1993

Design and analysis of real-time multimedia synchronization protocol on the high-speed transport layer

Hun Kang
Iowa State University

Order Number 9334992

Design and analysis of real-time multimedia synchronization protocol on the high-speed transport layer

Kang, Hun, Ph.D.

Iowa State University, 1993

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Design and analysis of real-time multimedia synchronization protocol on the high-speed transport layer

by

Hun Kang

A Dissertation Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department: Electrical Engineering and Computer Engineering
Major: Computer Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Members of the Committee:

Signature was redacted for privacy.

Iowa State University
Ames, Iowa
1993

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank my major professor, Dr. Douglas W. Jacobson for his insight, guidance, and encouragement for the past four years of my doctorate study. He helped me start this challenging research topic and guided me when I needed it most. His technical excellence and foresight has influenced me to pursue my future studies and researches.

I also would like to thank the professors of my committee for their help and guidance throughout my academic years. Dr. Johnny S. Wong in the Department of Computer Science taught important concepts of computer networks and operating systems in his lectures. Dr. Way Kuo, chair of the Department of Industrial and Manufacturing Systems Engineering helped my research despite his busy schedule. Dr. Terry A. Smay in the Department of EE/CprE always gave me a good advice for my work, and his lecture about information theory will be a good basis for my future research. Dr. Richard E. Horton in the Department of EE/CprE not only took great care of me since I started my study here but also helped me to get financial support to continue my studies. Working with him as a teaching assistant for two years extended my knowledge about the small computer systems. Also, I would also like to thank Dr. Gurpur M. Prabu in the Department of Computer Science for having attended my preliminary examination and giving me a good advice for the research.

I also would like to thank all the professors in the department and my friends who helped anytime and with whom I had a lot of good times. They all made me feel comfortable here, and I enjoyed sharing knowledge with them.

Finally I would like to thank my wife, Soyoung Joo, who encouraged me anytime I was in trouble and shared all the good and the bad with patience. She was also my best advisor at home. Without her endless support, nothing could have been possible for me. I thank my mother in Korea who always prays for me and my family. I also thank my son Kihan, who is my best friend at home. He came here as a baby and has grown up to be a wonderful child.

CHAPTER 1. INTRODUCTION

Concept

The speed of computer networks has been increasing at a dramatic rate. Current high speed networks (e.g., FDDI or B-ISDN) can handle more than 150 Mbps, and network reliability has remarkably increased (the bit error rate dropped from 10^{-6} to 10^{-9} and below) [1]. With high speed computer networks, it is possible to handle various types of data effectively, e.g., voice, video, data, and image. Many applications that need multimedia services are emerging [2].

In multimedia communication, several heterogeneous data types are transmitted and received, and multimedia data usually require real-time processing. When two or more media have certain relations, especially in the time domain, the temporal relations must be preserved (synchronized) during transmission so that the media retain the same relations at the destination to deliver meaningful information. Also, for real-time data in digital transmission, data packets in the same channel can have certain timing relationships. A conventional synchronization method provides a completely synchronized channel, e.g., TV broadcasting. The completely synchronized channel gives perfect combining among channels, and no further synchronization problems exist. This method has several problems when used in high speed data transmission [3] [4] [5]: 1) Jitter, 2) Using single QOS (quality of service), 3) Inefficiency, 4)

Source and sink complexity, and 5) Single origination for multimedia. To overcome these problems, a separate channel should be assigned to each medium, and some method must be provided to synchronize those channels. Also, it is believed that the synchronization facility should be incorporated in the transport layer. The transport layer is the minimal layer required by future applications, and the layer is certainly required for data communication since a reliable method of transmitting arbitrarily long data blocks by several users over a single channel will be a bare minimum service necessary.

In this research, a protocol to provide packet synchronization of several multimedia channels that is incorporated in a high-speed transport layer is designed and implemented. The transport layer that is incorporated with this synchronization method will be modeled, analyzed, and implemented to show acceptable results.

Why The Synchronization Problem Must Be Solved

1. Proliferation of distributed computing needs several heterogeneous data types.
2. Increased network capacity and reduced protocol processing time result in efficient handling of high-throughput data like multimedia.
3. Efficient use of the underlying network is required.
4. Multimedia communication naturally fits on future channels of B-ISDN.
5. No multimedia synchronization scheme is found in current standards.
6. No proposed synchronization scheme has been implemented or analyzed to give a satisfactory result.

Goals of the Research

The goal of this research is to design a transport protocol having multimedia packet synchronization function with following characteristics.

- A flexible packet synchronization protocol for multimedia streams on a high-speed communication environment.
- A scheme applicable for interactive real-time applications.
- A scheme suitable for current and emerging high-speed transport protocols, implementations, or their natural extensions.
- Effective handling of out-of-synchronization.
- System parameters that can effectively handle multimedia communication.

Design Approach Method

Real-time Synchronized Multimedia Transport Protocol (RSM-TP) is a transport layer protocol that has multi-channel synchronization capability. Its design is based on the current OSI transport protocol. The design approach of the protocol is as follows.

Selecting Multimedia The most common forms of multimedia data are voice, video, and text data. In this research, four data channels are used: 1) voice, 2) video, 3) synchronized data, and 4) independent data. Voice, video, and synchronized data channels are assumed to have certain timing relations among one another. Independent data is the normal data in OSI-TP.

Transport Protocol Implementation The general transport protocol is designed and implemented based on OSI transport layer protocol using OPNET network modeling/analysis package.

Designing Synchronization Protocol The synchronization protocol is designed using combined concept of synchronization marker and synchronization channel, which can handle various real-time timing relations efficiently.

Combining TP with Synchronization Protocol The designed synchronization scheme is combined with the transport protocol.

Simulation and Analysis OPNET simulation software is used to implement and simulate the designed protocol and to analyze the results.

Organization of the Dissertation

This dissertation is composed of eight chapters. Chapter 2 describes the brief review of current high-speed transport protocols, problems with the protocols, and multimedia communication using a high-speed network. Chapter 3 introduces concepts of multi-channel synchronization and proposes a synchronization scheme. Chapter 4 shows the design of the transport protocol with synchronization function based on the OSI transport protocol. It shows the network configuration, remote nodes, and the processes that are running in each node which generate desired protocol functions. Chapter 5 describes queueing models of the network and analyzes them. Chapter 6 describes simulation results from the OPNET network simulation tool. Chapter 7 describes the analysis of simulation results and suggests related future research. Finally, chapter 8 describes conclusions.

CHAPTER 2. HIGH-SPEED TRANSPORT PROTOCOLS FOR MULTIMEDIA TRANSMISSION

The primary function of any communication protocol is transmission of some meaningful data from one end user to another. In the communication protocol stack, the transport layer or its equivalent is the lowest layer that can handle user-to-user communication and the highest layer of the transport service provider as shown in Figure 2.1 [6] [7]. Any requirement and restriction of the meaningful transmission

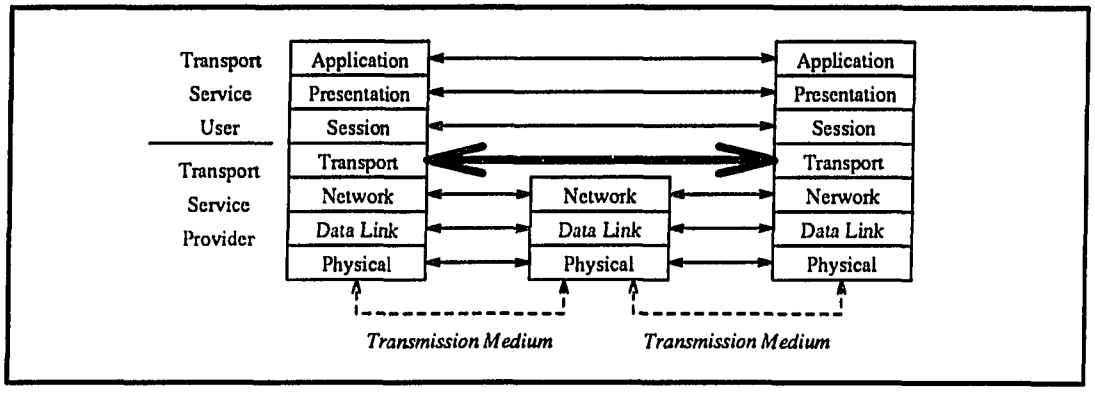

Figure 2.1: ISO OSI 7 Layer Reference Model

of data must be satisfied at the transport layer. Synchronization between several media is one of these requirements to be satisfied. A cross-stream synchronization scheme of multimedia data in the session layer is suggested in [8]. However, the synchronization has to be implemented in the lowest layer of end-to-end connection to

fully utilize existing protocol functions. As the communication speed becomes higher and higher, the system can handle more complex data in real-time. It can support high-throughput data transfer, multimedia application and multicast communication.

In this chapter, the OSI transport protocol is shown, emerging high-speed transport components are shown, problems of conventional transport protocols are mentioned, and multimedia communication over a high-speed transport protocol is explained.

Transport Layer Protocol

The transport layer provides transparent transfer of data between entities relieving the user of any concern with the detailed way in which reliable and cost effective data transfer is achieved. The services and functions of the OSI transport layer are listed below [9].

- Services
  - Transport connection establishment
  - Data transfer
  - Transport connection release
- Functions
  - Mapping transport addresses onto network addresses
  - Multiplexing transport connections onto network connections
  - Establishing and releasing of transport connections
  - End-to-end sequence control on individual connection

- End-to-end error detection and any necessary monitoring of the quality of service
- End-to-end error recovery
- End-to-end segmenting, blocking, and concatenation
- End-to-end flow control on individual connections
- Supervisory functions
- Expedited transfer

High-Speed Transport Component

Two primary approaches exist to achieve a high-performance transport component: 1) High-speed protocols and 2) Optimized implementations [10] [11] [12] [13] [14] [15]. Another approach is the design and implementation of a new multi-processor architecture [16].

1. High-Speed Protocols

   - TP5 (INRIA, France): TP5 is a transport protocol designed for the transport of real-time multimedia data in addition to normal data. The protocol is a variant of and compatible with OSI TP4. Flow control and error control are not performed on real-time data.
   - TP++ (Bellcore Lab): TP++ is a high-speed transport protocol for multimedia applications (1Gbps). Multiplexing of several transport connections is precluded to simplify the protocol design and to allow a more efficient support of QOS parameters.

- VMTP (Stanford University): VMTP (Versatile Message Transaction Protocol) provides transport communication between network-visible entities via message transactions which consist of a request message sent by a client entity to one or more server processes, followed by zero or more response messages sent back to the client by the server processes, at least one per server entity.
- XTP (Silicon Graphics): XTP (eXpress Transfer Protocol) is an approach to protocol design that uses a Protocol Engine (PE) concept.
- Delta-t (Lawrence Livermore Center): Delta-t was developed for a capability-based distributed operating system called LINCS. The protocol is designed for stream and transaction oriented communications and uses light-weight connection management. The major concern of Delta-t is connection management, and the transmission is based on a timer mechanism.
- NETBLT (MIT): NETBLT (Network Block Transfer Protocol) was developed for high-throughput bulk data transfer. Separating data and control flow allows an efficient independent implementation of both.
- HOPS (AT&T Bell Labs): HOPS (Horizontally Oriented Protocol Structure) divides protocol into functions instead of layers. The functions are mutually independent and can be performed without knowing the results of execution of another, thus making multi-processing possible.

2. High-speed Implementations

   - VLSI
     - Protocol Engine (Silicon Graphics): PE (Protocol Engine) is a hardware architecture for implementing network protocols and system interface using VLSI techniques.
     - PSi Compiler (IBM Yorktown): PSi is a silicon compiler that transforms formal protocol specifications into efficient VLSI implementations. It takes advantage of the parallelism intrinsic to a given protocol to accomplish very high-speed implementation.
     - Translation to VLSI (AT&T Bell Labs)
     - MCM (IBM Yorktown): MCM (Modular Communication Machine) aims at a modular system design based on functional units (e.g., protocol function) that are programmable for special protocol requirements.
   - High-Speed Adapter Board
     - NAB (Stanford University): NAB (Network Adapter Board) is specially developed for the VMTP and is optimized for working on VMTP messages (e.g., calculation of checksum on the fly).
     - Petrinet Controller (Aachen University): Petrinet Controller is designed for realizing petri networks that describe the protocols.

3. Multi-processor implementation (Transputer): The following protocols are implemented by building a global memory for Transputers for high performance protocol implementations.

- OSI LLC Protocol (IBM Rüschlikon)
- OSI LLC Protocol (Erlangen University)
- OSI Layer 3 and 4 (Karlsruhe University)

Problems with Transport Protocols and Their Implementations

Conventional transport protocols have several problems that must be solved for real-time high-speed multimedia applications [4] [17].

1. Most implementations are tied heavily into host operating systems, making the protocol processing slow.
2. They were designed with different constraints and requirements than those needed for high-speed applications.
3. They were designed to minimize the number of bits transmitted to reduce insertion time, which results in complex decoding of packets.
4. Packet field limitation (e.g., sequence number).
5. None of the synchronization, multiconnection management, and multicast facilities are currently defined, which are necessary for multimedia applications.

Among these problems, synchronization is one of the important issues to be addressed for a multimedia environment.

Multimedia Communication

Several underlying technologies have been developed for high-speed communication, including B-ISDN, QPSX, FDDI, and fast packet switching. As the network

speed becomes higher, multiple simultaneous data streams need to be transmitted through the network. These higher capacity networks and increased processing capability are able to support multiple voice, video, and other data streams simultaneously, thus allowing for more complex communication patterns than single media point-to-point communication. It is also no longer necessary to build entire workstations specifically to handle the multimedia stream efficiently, thus leading to a desire for open systems [3]. An open system is one which can be incrementally extended by the addition of new functionality without disturbing the existing system components. The transport service must be extended and the QOS (Quality Of Service) parameters must be adjusted as follows [5].

- Selection of different data rates or dynamic request of bandwidth
- Quality selection in respect to guaranteed throughput, transmission delay, security, error identification, and error handling
- Combination of isochronous and anisochronous communication

The isochronous data has the characteristic of a time scale or signal such that the time intervals between consecutive significant instants either have the same duration or durations that are integral multiples of the shortest duration. In isochronous data the variation in time intervals must be kept to a minimum.

Information in a form accessible to humans is generally associated today with the term **multimedia**, and multimedia systems can be defined as systems capable of handling a wide variety of information formats, such as text, voice, graphics, image, audio, and video data [2]. The multimedia stream usually contains voice and video components as well as the normal data and requires high-throughput networks. The

real-time voice and video data streams are isochronous in nature, that is, they can be thought of as a stream of finite sized samples which are generated, transmitted, and received at fixed time intervals, imposing a set of timing constraints which must never be exceeded. If the source, network, and sink ran completely synchronously, without errors, and introduced no queueing delay, then the source and sink would always remain in synchronization and there would be no need for buffering at the sink. But in the real network, there always are statistical queueing delay and errors, which make the sink do some buffering of the packets. The buffer is also necessary to maintain the timing constraints among the different channels. Followings are examples that need multimedia communication capacity [18].

- Multimedia electronic mail
- Real-time conferencing
- Cooperative design
- Medical application
- Distance learning

These multimedia streams may contain voice, video, compressed video, data, real-time data, and still image, and some means of controlling and synchronizing these multiple streams are required. Also, some components of the multimedia streams have to be processed in real-time. These real-time constraints limit timing parameters, and must allow the error rate to a certain degree to compromise with the real-time data processing.

Table 2.1: QOS Parameters for Real-time Multimedia Data

QOS Parameters	Maximum Delay (s)	Maximum Delay Jitter (ms)	Average Throughput (Mbps)	Acceptable Bit Error Rate	Acceptable Packet Error Rate
Voice	0.25	10	0.064	$< 10^{-1}$	$< 10^{-1}$
Video	0.25	10	100	10^{-2}	10^{-3}
Compressed Video	0.25	1	2 - 10	10^{-6}	10^{-9}
Data	1	-	2 - 100	0	0
Real-time Data	0.001 - 1	-	< 10	0	0
Image	1	-	2 - 10	10^{-4}	10^{-9}

These parameters must be passed to the transport layer for proper processing of data. Table 2.1 shows their required QOS (Quality Of Service) parameters for meaningful communication [19]. These parameters must be satisfied through end-to-end communication.

The values vary considerably according to the data types, and they result in the necessity of separating the channels between data types. The timing problem of the separate channels can be solved by the protocol designed in this research.

CHAPTER 3. CONCEPTS FOR MULTIMEDIA SYNCHRONIZATION

Synchronization in the context of multimedia refers to a mechanism used by the processes (also specific to multimedia) to coordinate their ordering in the time domain [20]. There are mainly two places in which the synchronization is required: 1) Retrieval of the stored multimedia data and 2) Receiving multimedia data from a remote source using different communication paths for different media. The stored multimedia data usually have several sources and the data are used by several different devices on retrieval. A technique for synchronization of multimedia at the time of storage, and retrieval from network file servers was designed in [21]. In this research, the multimedia synchronization between two remote users in the communication network is focused.

Why Does The Synchronization Problem Occur

For the distributed processing of multimedia information, it is necessary not only to integrate isochronous and anisochronous communications, but to combine several types of isochronous data (such as video and voice) in order to maintain their relations from one end-user to another. There are two types of conceivable synchronization methods for the separate channels as shown in Figure 3.1. For the low speed com-

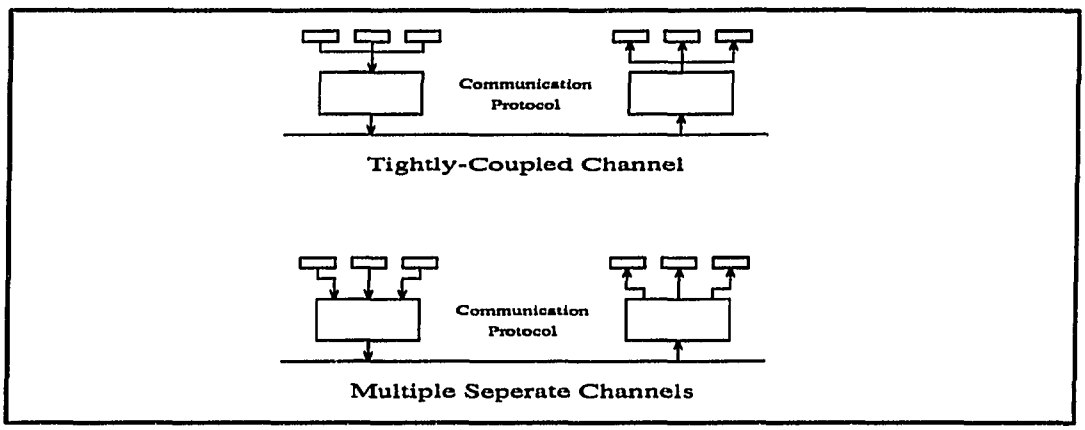

Figure 3.1: Multimedia Communication Channels

puter communication network, usually one type of data is transmitted and received, in which nothing is to be synchronized. Conventional multimedia transmission uses a tightly synchronized channel, which needs no further synchronization. For example, in TV broadcasting, video and sound are sent through the same channel. If the multimedia communication is used with the high-speed computer communication, several limitations and requirements are to be considered:

- Interactive communication with multimedia is necessary.
- The media use complex communication protocol stacks (e.g., OSI reference model).
- Protocol processing time is different for each media.
- Different requirements (e.g., QOS) are applied for different media.
- Communication capacity (protocol processing and network capacity) is limited.

Due to the above reasons, using a single channel for multimedia results in inefficiency of network utilization, complexity of source and sink, and it requires single origination for multimedia. Therefore, for multimedia communication, a separate channel is dedicated to each medium, and the media are combined at the destination. When the multimedia informations arrive at the destination, there is no guarantee that the information elements retain the same order as that at their source. Also, for the high-speed network to improve system performance, the total network bandwidth may be divided and several subchannels are used in parallel [22], in which case the timing relations are easily lost during transmission. The order in one channel can be maintained by the sequencing function in the transport layer, but the timing relation between packets in one channel or among several channels is not maintained by the current transport protocol. A certain function of synchronization must be performed on the information streams on the separate channels.

Followings are common examples that need synchronization on the multimedia communication environment.

Lip-synching [3] This refers the synchronization of spoken voice with the movement of speaker's lips.

Document Transmission [4] ODIF (Office Document Interchange Format) standard specifies the contents of IDEs (Interchange Data Elements) and the order in which they must be converged. However, it does not specify that all the elements must be transmitted together in a block. It allows separate transmission of text, data, image, and structure information, which results in parallel transmission.

Relation Between Multimedia Information

Each multimedia data are sent through separate networks and those data are coordinated (synchronized) in the transport layer or its equivalent layer at the destination. The multimedia data packets can have three kinds of relation each other, called presentation control [4]: 1)Simultaneous, 2) Sequential, and 3) Independent.

Simultaneous The packets are to be sent to their users at the same time. An example is the video accompanied with its corresponding voice.

Sequential One packet is to be sent after the other packet has completely been sent to its user, or vice versa. An example is sending several slides. Each slide has its order and the order must be kept at the destination.

Independent Either packet can be sent to its user at a random rate. Usually the first arrived packet is processed first.

Those presentation control can be nested, for example, any simultaneous group of data can have sequential relation with the other simultaneous group. An example of this presentation control is shown in Figure 3.2.

Object Composition Petri Net (OCPN)

More formal specification and modeling of processes that have temporal relation can be found in [23], in which any two atomic processes specified by temporal intervals are modeled using Object Composition Petri Network (OCPN). There exist seven relations that can express the two processes. Those relations can also be expressed in one unified OCPN model. Figure 3.3 and Table 3.1 show the unified OCPN model

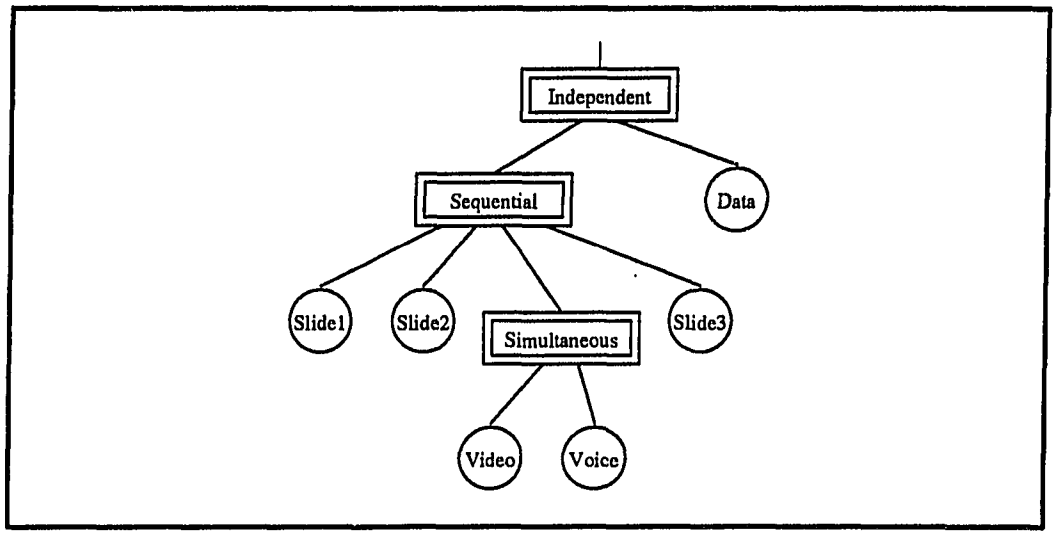

Figure 3.2: Multimedia Presentation Control

and its temporal parameters for the seven relations. The OCPN model can be nested to represent more complex timing relation.

Synchronization on Communication Channels

Little research has been done on how to get proper synchronization between multimedia streams over communication channels. A multiple virtual circuit method has been proposed [24]. Two conceptual methods (synchronization marker and synchronization channel) are also mentioned, and OSI extension of the synchronization marker is proposed [4].

Multiple Virtual Circuit

A multiple virtual circuit multiplexes packets of different information streams into a sequence of a packets while providing a multiple information channel interface

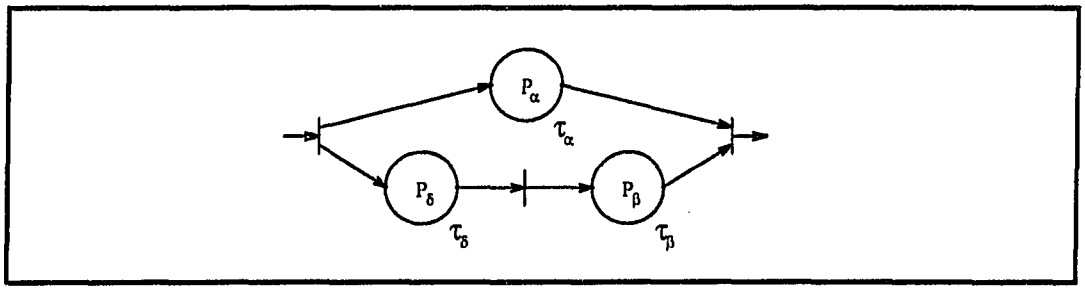

Figure 3.3: Unified OCPN Model

Table 3.1: Temporal Parameters for Unified OCPN Model

Relation	Parameter Relationships	Overall Duration
P *before* P_b	$\tau_d \neq 0$	$\tau_{TR} = \tau_a + \tau_b + \tau_d$
P_a *meets* P_b	$\tau_d = 0$	$\tau_{TR} \geq \tau_a + \tau_b$
P_a *overlaps* P_b	$\tau_a < \tau_b + \tau_d$, $\tau_d \neq 0$	$\tau_{TR} = \tau_b + \tau_d$
P_a *during*$^{-1}$ P_b	$\tau_a > \tau_b + \tau_d$, $\tau_d \neq 0$	$\tau_{TR} = \tau_a$
P_a *starts* P_b	$\tau_a < \tau_b$, $\tau_d = 0$	$\tau_{TR} = \tau_b$
P_a *finishes*$^{-1}$ P_b	$\tau_a = \tau_b + \tau_d$, $\tau_d \neq 0$	$\tau_{TR} = \tau_a$
P_a *equals* P_b	$\tau_a = \tau_b$, $\tau_d = 0$	$\tau_{TR} = \tau_a$

to upper layer software [24]. At the source, the transport layer receives multiple information streams from its user, multiplexes them into a packet, and sends it to the network layer as in Figure 3.4.

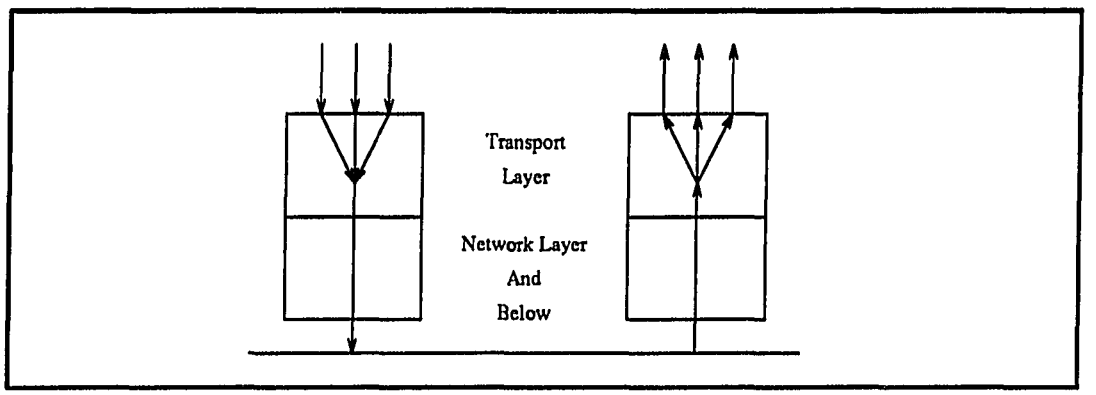

Figure 3.4: Multiple Virtual Circuit

This scheme does not generate further synchronization problems, but it suffers the same problem as in the completely synchronized channel.

Synchronization Marker

The synchronization marker (SM) method is to send a separate packet containing synchronization information prior to the data packets to be synchronized to inform the destination process to take proper actions to synchronize them. This scheme is shown in Figure 3.5. The SM method gives the simplest form of synchronization, but it has following problems:

- Large buffer is required at the destination.
- The data stream is modified by the transport system.

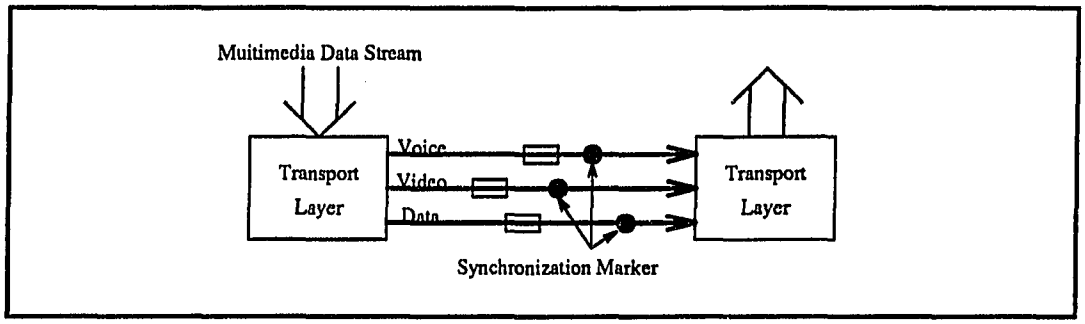

Figure 3.5: Synchronization Marker Scheme

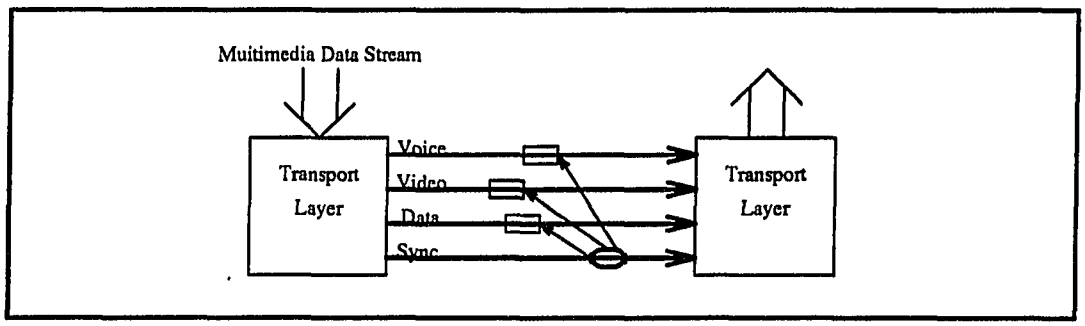

Figure 3.6: Synchronization Channel Scheme

- Only parallel synchronization is supported.
- An extra TPDU must be made and sent as an SM packet.

Synchronization Channel

The synchronization channel (SC) method uses another channel to carry the synchronization information as in Figure 3.6. The synchronization information can be any type of timing information (parallel, series, or any order). In this scheme, an extra channel needs to be used even for simple synchronization (e.g., parallel).

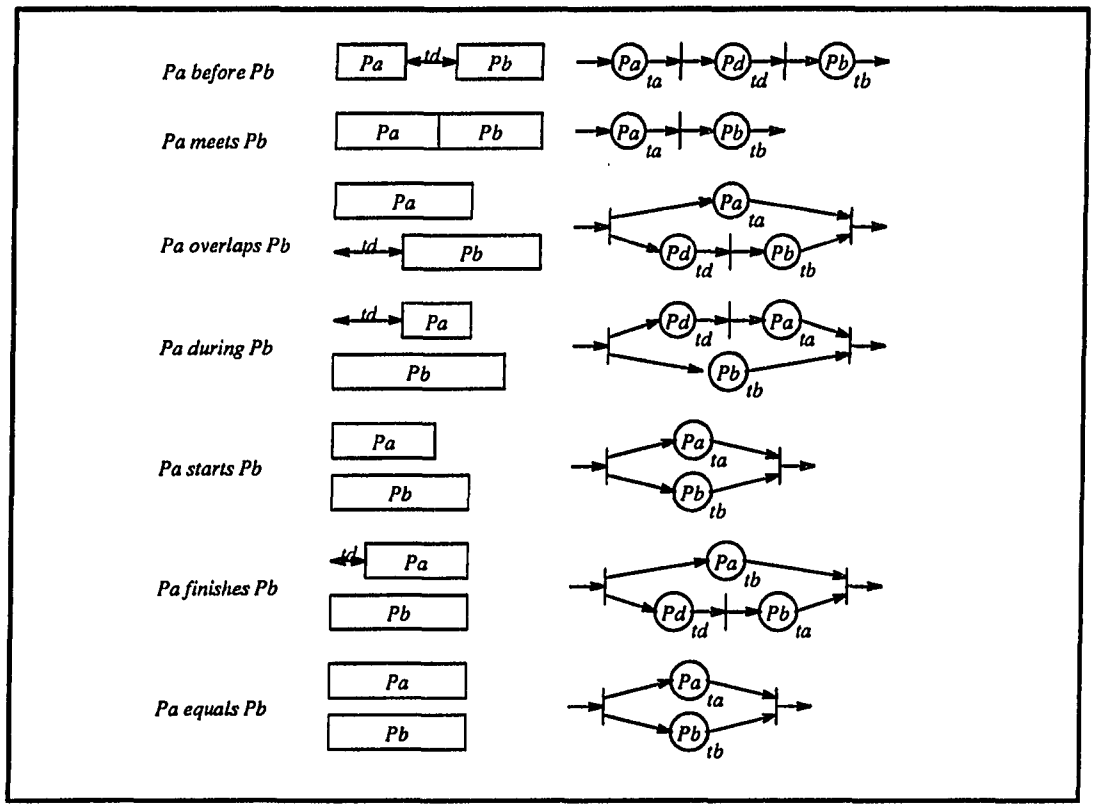

Figure 3.7: Temporal Relations and Their OCPN

Designed Synchronization Scheme

The synchronization scheme designed in this research uses the seven temporal relations shown in [23], and is to be operated in the transport layer. Actually there can be thirteen temporal relations if the reverse relations are to be considered. There is no reverse relation for *equal* relation. The reverse relations can be treated with the same manner. The seven temporal relations and their corresponding OCPNs are shown in Figure 3.7. P_a and P_b can be thought as data packets (TSDU) on the transport layer. Each data packet is sent to the user of the transport service at the destination. Since the time of the end of the packet is not very meaningful to the

user, only the start timing is of importance. Whenever the synchronization condition is satisfied, the related packets are sent to the user as fast as possible. Therefore the seven temporal relations can be reduced to four relations: *before*, *meet*, *overlap*, and *equal*. *Before* and *meet* are serial synchronization, and *overlap* and *equal* are parallel synchronization.

The goals of the designed synchronization scheme are:

1. Provide a simplest scheme for *equal* synchronization, which guarantees the least network overhead.
2. Provide a flexible scheme for the other types of synchronization.
3. Generate acceptable overhead for protocol processing and underlying network.
4. Fit on the natural extension of current emerging high-speed transport protocols or implementations.
5. Suitable for interactive real-time communication.

EQUAL Synchronization

While this type of synchronization is most frequently required, it is important to make it simple and not consume many network resources. In an SM scheme a separate SM-TPDU which contains a sequence number and connection numbers that are to be synchronized is sent. This method needs to send an SM-TPDU prior to sending every data packet that needs to be synchronized. In the designed scheme of this research, instead of using a separate synchronization marker packet, the synchronization information is included in the data packet (DT-TPDU), so that the synchronization

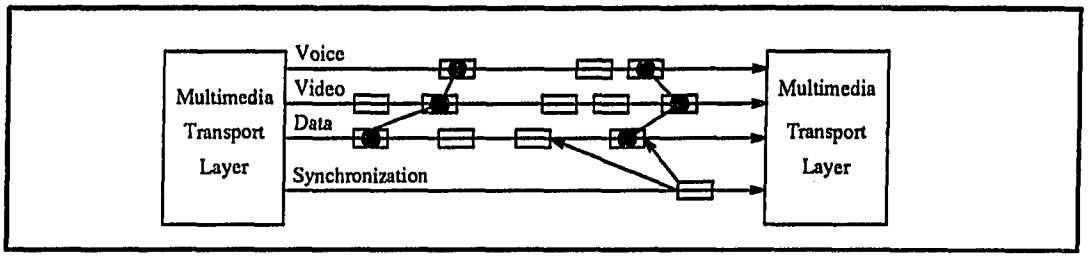

Figure 3.8: Combined Synchronization Scheme

information can be decoded during normal DT-TPDU decoding. The variable header part in DT-TPDU can be used to store the synchronization information. Currently the only parameter defined in the variable header part of OSI DT-TPDU is checksum.

Non-EQUAL Synchronization

One or more connections are made to carry more complex synchronization information. This channel carries the information about the channels to be synchronized, sequence numbers of the packets to be synchronized, delay parameters, and any other necessary parameters. The packet format over the synchronization channel needs to be specified. This scheme is to be designed to specify any type of temporal relations with small network overhead.

COMBINED Synchronization Scheme

The above two synchronization schemes are combined to give a scheme that can handle both methods as in Figure 3.8. In this COMBINED synchronization scheme, synchronization information is carried in both places. The synchronization information in the DT-TPDU takes care of the simplest synchronization (EQUAL). Other types of synchronization information are carried on the separate channel only

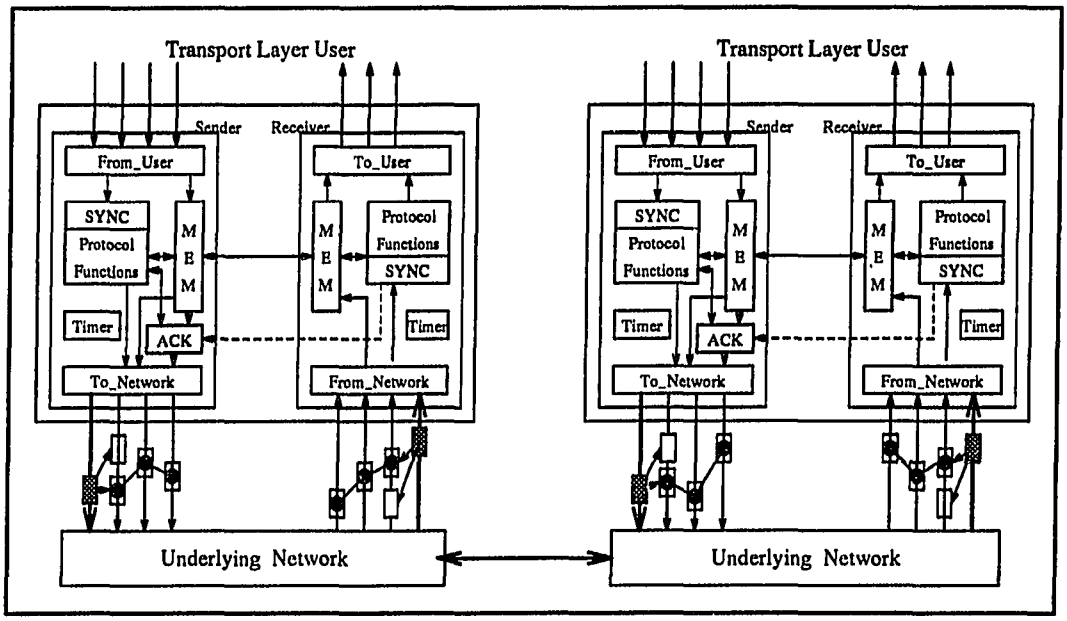

Figure 3.9: Model of Transport Layer with Synchronization

when they are needed. This designed scheme is expected to provide the maximum synchronization capacity with minimum overhead.

Transport Protocol Model

The conceptual end-to-end transport layer model with the proposed synchronization protocol is shown in Figure 3.9. Among several emerging high-speed transport protocols, The OSI transport protocol was chosen for this protocol design for the following reasons:

- OSI transport protocol has well-defined standards.
- OSI protocol is felt to be a dominating protocol [25].

- Recent research shows that OSI TP4 with general purpose multiprocessor (transputer) implementation can get 60 Mbps throughput, and would be faster by using special purpose processors [26].
- An efficient multi-processing architecture and accompanying data structures for parallel processing of protocol packets with minimum contention have been shown such that a steady state throughput of 1 Gbps is achievable with modest number of packet processors [17].
- Putting protocols in hardware is not necessarily required for high-speed implementation and suitable network interface will be needed to use standard protocols [27].

The transport layer receives the data request for each media and SYNC request from its user. The protocol machine performs the normal transport protocol operations and also performs the appropriate synchronization procedure. At the receiver side, the transport layer receives the data and synchronization information from the network layer, synchronizes them, and sends the synchronized data to their users. To incorporate this synchronization scheme, the transport state diagram is modified by inserting states for synchronization actions.

Out of Synchronization

The DT-TPDU with synchronization information can be lost or delayed, which results in out of synchronization. Also the synchronization packet in the synchronization channel can be lost or delayed too. Non-real-time data usually requires an error correction procedure. But in real-time situation, retransmitting packets is usually

not permitted. Therefore where error recovery is not possible, errors are reported to the user to take a proper action. When the synchronization effort in the real-time multimedia fails, a behavioral concept called **Restricted Blocking** is proposed [20]. In the restricted blocking, when the synchronization of streams fails, the waiting multimedia object is not suspended, instead it performs an alternative action. This action is aborted as soon as the synchronization condition rises.

CHAPTER 4. DESIGN OF RSM-TP

The Real-time Synchronized Multimedia Transport Protocol (RSM-TP) is designed to be a natural extension of the existing OSI transport protocol. The RSM-TP includes a multi-channel transport protocol with synchronization functions among channels. This protocol is designed on a DEC 3100/5100 workstation, using OPNET network modeling tool which provides comprehensive network modeling, simulation, and analysis environment [28]. The detailed flowcharts of the RSM-TP are given in Appendix.

Synchronization Scheme Design

The goal of this design is to make a transport protocol that can handle various synchronization tasks with minimum overhead. The protocol receives $T\_SYNC.req$ from the user with synchronization information. The protocol analyzes the request and performs proper synchronization functions. Also, if the protocol receives the synchronization information from the network, it analyzes the information and synchronizes the received data packets when they are sent to their users.

If $T\_SYNC.req$ comes at time t from the transport service user, let the packet coming from its user of channel A after the time t be A_i, where $i = 1, 2, 3, \cdots$, and A_i comes earlier than A_j if $i < j$. Three types of synchronization schemes are

designed: 1) EQUAL, 2)Non-EQUAL, and 3) COMBINED.

EQUAL The specified packets in the three real-time channels are delivered to their remote users at the same time. If the channels are A, B, and C, the packets A_1, B_1, and C_1 use the DT-TPDU containing EQUAL synchronization information.

Non-EQUAL The two packets are delivered to their remote users with a specified time delay (t_d). If the two channels are A and B, and $A \neq B$, A_1 and B_1 are the specified packets. If packet A_1 is transmitted and delivered to its user at the time t_{a1}, the packet B_1 is delivered to its user at $t_{a1} + td$. If the two channels are the same channel (A), A_1 and A_2 are the specified packets.

COMBINED This is the combined scheme of the EQUAL and the Non-EQUAL synchronization. The packets A_1, B_1, and C_1 are delivered to their users at the same time (t_{a1}), and the next packet in the specified channel (A_2, B_2, or C_2) is delivered to its user at the time $t_{a1} + t_d$.

Figure 4.1 and Figure 4.2 show simplified diagrams of the transport protocol state transitions with synchronization for sender and receiver respectively.

Network Design of RSM-TP

The RSM-TP is intended for two distinctive users to communicate with each other using multi-channel data that have important temporal relations among the packets. The communication network with two users is modeled as in Figure 4.3. TP1 and TP2 have symmetric functions, and each transport entity accepts multime-

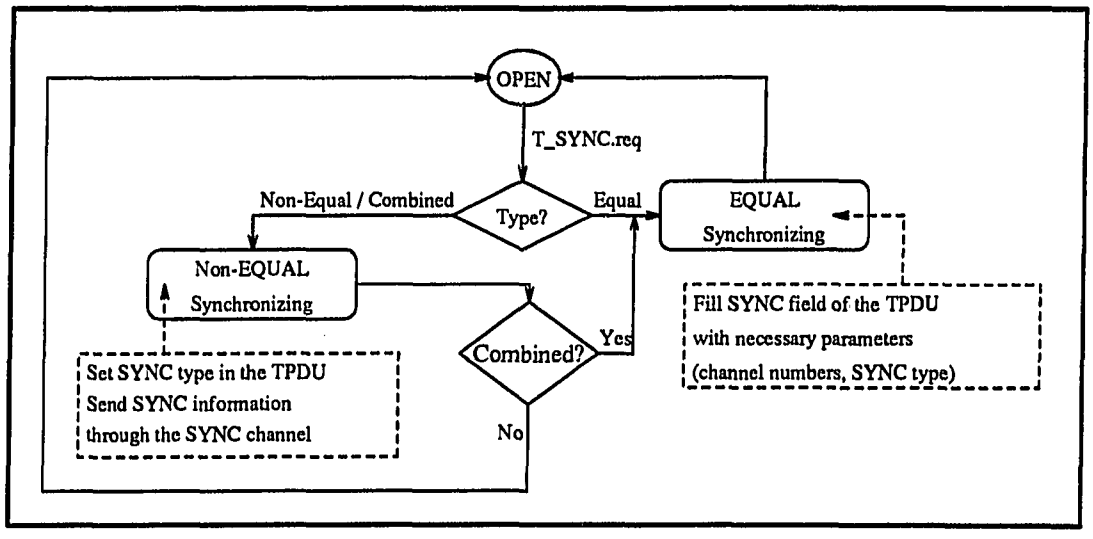

Figure 4.1: State Diagram for Sender

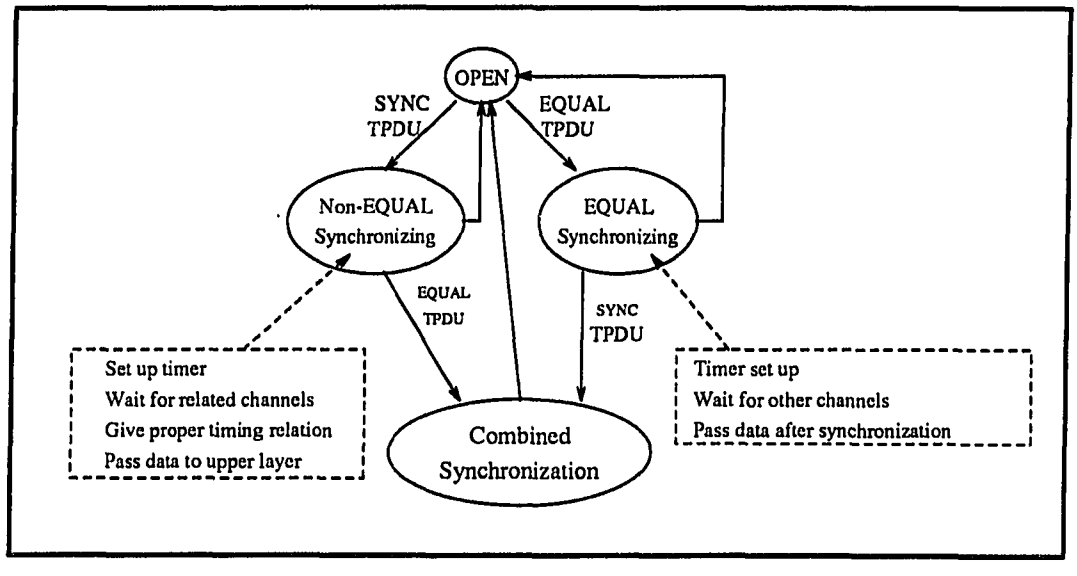

Figure 4.2: State Diagram for Receiver

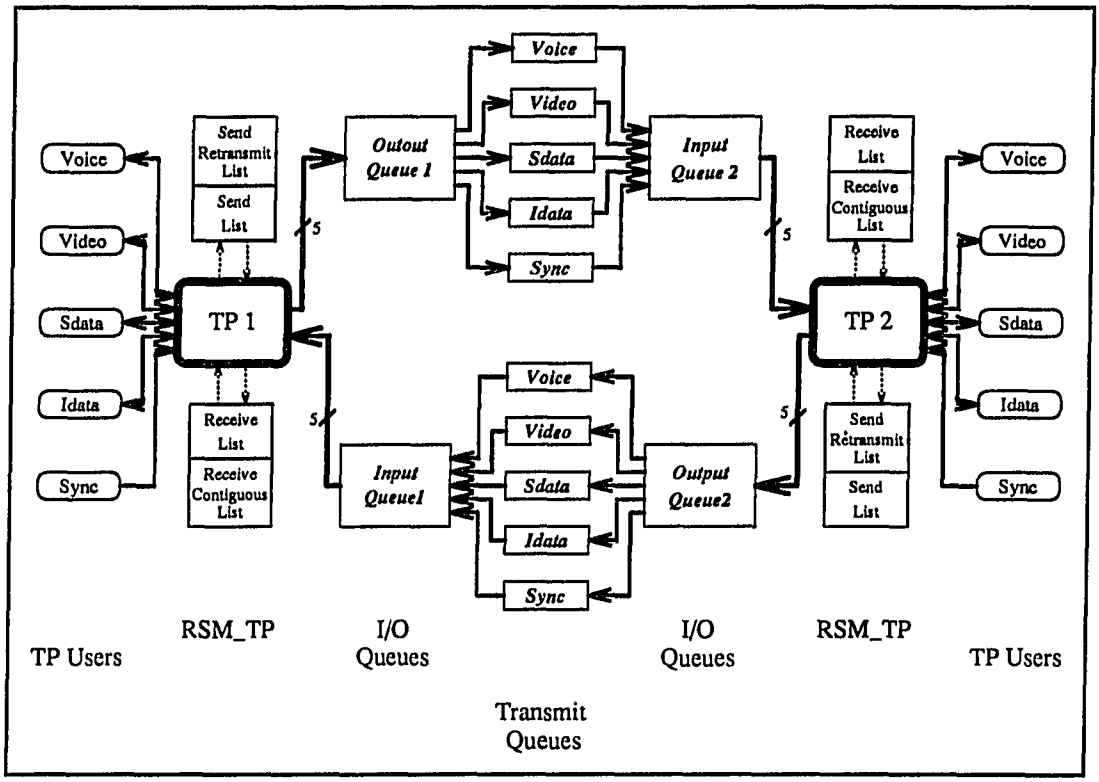

Figure 4.3: RSM-TP Network Configuration

dia data from its user with timing information. If no T_SYNC_{req} is given by its user, the transport protocol works as if it were a normal OSI transport protocol.

At the sender side, any given timing information is analyzed inside the transport protocol. The synchronization information is inserted into the DT-TPDU and/or separate synchronization packet (SYNC-TPDU) is sent through the SYNC channel. At the receiver side, the incoming packets are processed and passively passed to the user until any SYNC information arrives. The SYNC information can be obtained from the DT-TPDU or from the SYNC-TPDU through the separate SYNC channel. The transport protocol has four lists (buffers) that work as packet queues in the transport entity, and the size of the list is used to estimate resource capacity. The Receive Contiguous List (RCL) is mainly used for synchronization buffer. All the packets with proper sequence numbers remain in the RCL if they wait for any synchronization condition.

Transport Node Design

The transport node is configured as in Figure 4.4. Each channel data is generated from its packet generator, in which the packet size and the inter-arrival time can be determined during simulation. The transport service users (voice_proc, video_proc, sdata_proc, idata_proc, and sync_proc) get data from their generators, and send the data (TSDU) to the transport protocol processors (rsm_transport) with necessary interface information. These data are applied to the transport protocol which processes the packets and sends to the output queue (out_queue). The queue is a FIFO queue which simulates processing delay in the protocol. Each data element is sent to its own independent channel in the output transmitter (mult_out). The incoming

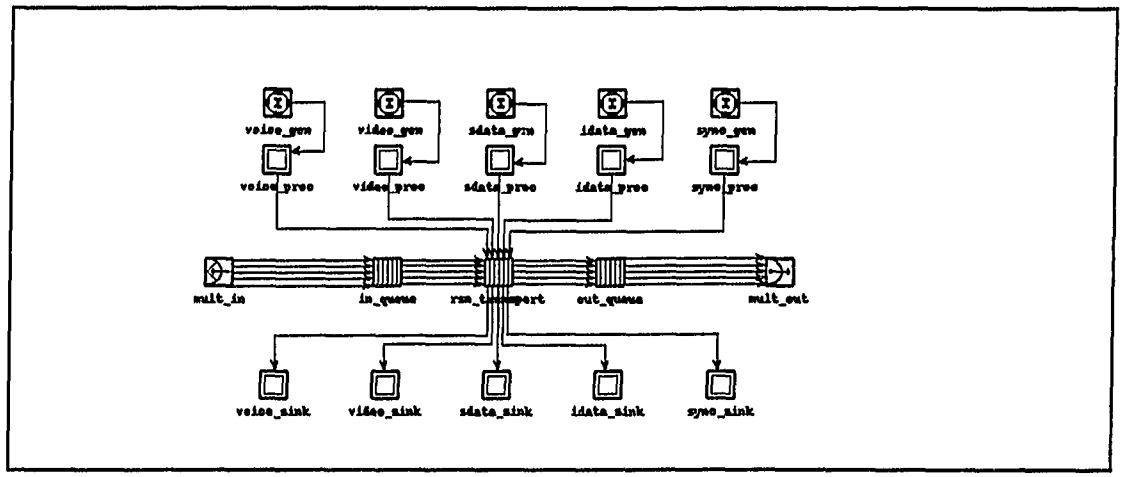

Figure 4.4: RSM-TP Node Configuration

data from the receiver (mult_in) are processed in the rsm_transport and sent to their own users. The mult_in and and mult_out are a point-to-point multi-channel receiver and transmitter. The data sent from one channel of the mult_out arrive at the corresponding channel of the mult_in. The transmission rate can be adjusted to get the desired speed of each channel.

Each node is composed of a process that has appropriate protocol functions. The process is designed using the process editor of the OPNET package. The process is represented as a state transition diagram. The functions in the state are implemented with the C programming language.

RSM_transport

The rsm_transport node contains the transport protocol which is the central part of this design. It is implemented in the node rsm_transport of Figure 4.4. The RSM-TP is mainly composed of four states: 1) INIT, 2) ESTAB, 3) SEND, and

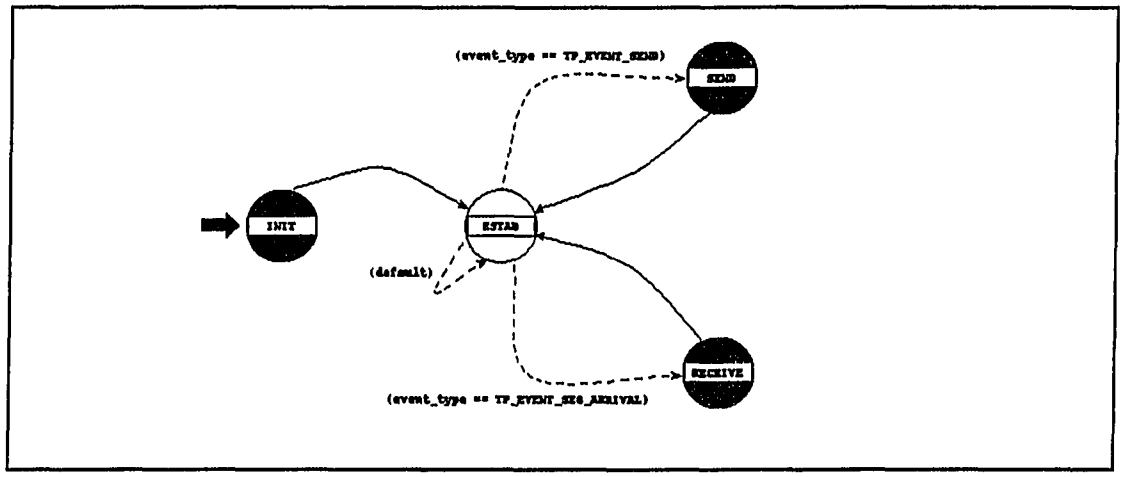

Figure 4.5: RSM-TP State Transition Diagram

4) RECEIVE. The state transition is shown in Figure 4.5. When the protocol is started it enters into the INIT state. After necessary initialization it enters the ESTAB state. The RSM-TP assumes the transport connections have already been made, and it considers the operation for the established connections. Among several events inside the transport protocol, send and receive events make the transition to the other states in the diagram. The other events (timeouts) are processed in the service routine inside the ESTAB state.

The followings are the major functions used inside the rsm_transport process.

tp_reset() Reset state variables.

tp_receive_reqs_process() Retrieve arrived packets, synchronize, and send the packets to users.

tp_send_reqs_process() Prepare packets with synchronization information.

tp_event_analyze() Analyze packet arrivals, and timer interrupts.

tp_enqueue_receive() Get arrived packets and insert them into receive queue.

tp_enqueue_send() Insert packets to be sent into send queue.

tp_enqueue_send_sync() Enqueue SYNC packets.

tp_segment_send() Send packets to lower layer.

tp_extract_contig_segs() Extract contiguous packets from receive queue.

sync_table_empty() Check if SYNC table is empty.

set_sync_table_send() Fill SYNC table with given information at sender side.

get_sync_table_send() Get current status of SYNC table at sender side.

tp_retrans_list_flush() Flush unnecessary packets from retransmit list.

tp_enqueue_ack() Insert ACK packets into send queue.

tp_send_to_user() Send packets to user.

In_queue, Out_queue

The in_queue is a FIFO (First-In-First-Out) queue that receives packets from the point-to-point receiver. The data coming from each channel is enqueued into the in_queue and sent to the transport protocol process. The outgoing data from the transport process is sent to the point-to-point transmitter through the out_queue. When the queue receives a packet, it determines the service time according to the packet length and the service rate of the queue. A timer is set to be expired at t, where $t = current\,time + packet\,service\,time$. At the same time, a busy flag is set to

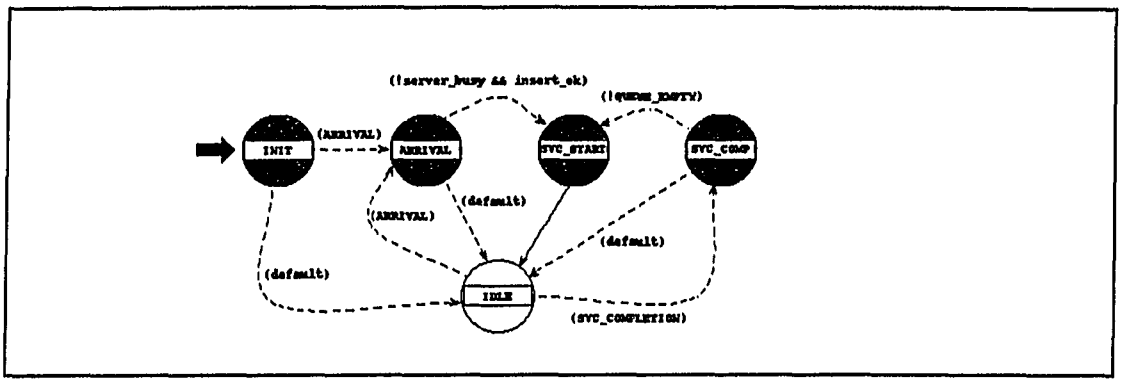

Figure 4.6: In/Out Queue State Transition Diagram

1 not to start another service. At the time t, the packet is dequeued and sent to its corresponding channel, and busy flag is reset, so that the next packet can be served. The enqueued packet waits or gets served immediately according to the busy flag. Both queues are functionally identical and the state transition diagram of the queues is shown in Figure 4.6.

User Process (data_proc)

The user process works as a transport service user. It gets data packet from the packet generator and sends the data to the transport process. Figure 4.7 shows the user process. For the SYNC user, the user process determines the type of synchronization. the channels to be synchronized, and the timing value. The request is sent to the rsm_transport as interface information between layers.

Mult_in, Mult_out

The mult_out is a point-to-point multi-channel transmitter and mult_in is a point-to-point multi-channel receiver. Each packet sent through a channel of the

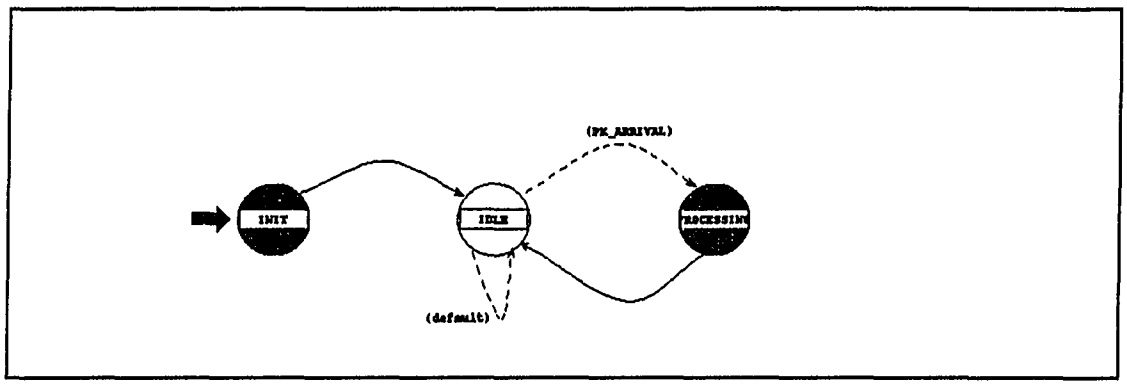

Figure 4.7: User Process State Transition Diagram

transmitter is received by the corresponding channel of the receiver. The point-to-point receiver is designed to generate errors for the incoming packets with bit error rate parameter. It generates the number of errors in a packet, and if it is greater than the threshold specified for each channel, the packet is discarded. The following algorithm is used to generate the number of errors in a packet [29]. Let

$$
\begin{aligned}
l &= \textit{packet size} \\
p_e &= \textit{Prob}\,[\,\textit{Bit Error}\,] \\
p_c &= \textit{Prob}\,[\,\textit{Bit Not Error}\,] = 1 - p_e \\
p_0 &= \textit{Prob}\,[\,0\ \textit{Errors In A Packet}\,] = {p_c}^l \\
p_n &= \textit{Prob}\,[\,n\ \textit{Errors In A Packet}\,] \\
p_a &= p_0 + p_1 + \ldots + p_i \ \ (0 < i < l - 1) \\
c_n &= \textit{Prob}\,[n\ \textit{Errors Occur} \mid \textit{At least}\ n-1\ \textit{Errors Occurred}] \\
n\_err &= \textit{Number of errors in a packet}
\end{aligned}
$$

Initially let $p_n = p_0$, $p_a = 0.0$, and $n\_err = l$.

$$
\begin{aligned}
&\forall\, i,\ \ i = 0\,..\,l-1 \\
&\quad c_n = \frac{p_n}{1.0 - p_a} \\
&\quad if\ \ (uniform\_distribution[0\,..\,1] < c_n)\ \{ \\
&\qquad n\_err = i \\
&\qquad break \\
&\quad \} \\
&\quad else\ \{ \\
&\qquad p_a = p_a + p_n \\
&\qquad p_n = (p_e)^{i+1}\,(p_c)^{l-(i+1)} \times \frac{l!}{(i+1)!(l-(i+1))!} \\
&\quad \}
\end{aligned}
$$

The n_err value is the number of error bits in a packet.

Packet Generator (data_gen)

The ideal generator module built in the *OPNET* package is used to model the packet generators. The ideal generator module is a stochastic packet source whose output can be shaped by specified probability distributions. These distributions can control the frequency of packet arrivals, and the length of packets. There are five packet generators that generate packets for each channel. The packet generation parameters used by each channel are summarized in the Table 4.1. The packet size of the real-time data has constant distribution since the data are thought of constantly sampled packets from the source. For the interarrival time, Voice and Video are given

Table 4.1: Packet Generation Parameters of Ideal Generator

	IA PDF [a]	IA Mean (secs)	PS PDF [b]	PS Mean (bits)
Voice	C [c]	2.000×10^{-3}	C	128
Video	C	8.192×10^{-5}	C	8192
Sdata	U [d]	1.024×10^{-3}	C	1024
Idata	E [e]	8.192×10^{-4}	E	8192
SYNC	U	1.500×10^{-2}	C	120

[a] Inter-arrival PDF
[b] Packet Size PDF
[c] Constant
[d] Uniform
[e] Exponential

constant distribution. Since Sdata has conceivable variation with some limit, it is given uniform distribution with mean 1.024×10^{-3} secs ($0.512 \times 10^{-3} \sim 1.536 \times 10^{-3}$ secs). SYNC request interval is also given uniform distribution with mean 1.5×10^{-2} secs ($1.0 \times 10^{-2} \sim 2.0 \times 10^{-2}$ secs). An experiment at CCETT (France) has shown that some applications require picture and sound to be synchronized within 150 ms [20]. The synchronization interval for this design is average 15 ms which gives enough synchronization frequency compared with the CCETT result.

Sink (data_sink)

The sink works as a transport service user at the receiver side. This node receives packets from the transport process and discards.

Flow Control

The RSM-TP adopts a credit scheme with strict acknowledgement, which acknowledges every coming data packet. At the receiver side, the received packet is inserted into the Receive Contiguous List (RCL) if the packet has an expected sequence number. The packet with a smaller sequence number than expected is duplicated packet and is discarded. If the sequence number is greater than expected, then the previous packet is missing or has not arrived yet. In this case, the packet is queued into Receive List (RL). This packet will be moved to the RCL if the earlier packet arrives or the timer expires. The acknowledge is sent when the packet is queued into RCL. The acknowledge number is the next expected packet number in the RCL, and the available number of empty space of RL is sent as a credit.

Error Control

The error is checked at the receiver. If the rate of error in a packet is greater than the error threshold, the packet is discarded. Every packet queued into the RCL schedules a timer, the expiration of which means that the following packet has been lost. The timeout increases the next expected packet number, and checks the RL to see whether other packets can be moved into RCL. At the sender, a copy of the transmitted packet is inserted into the Send Retransmit List (SRL) and a timer is set for the packet. If the timer expires, another packet is transmitted again. This is repeated for a designated amount of times and gives up. The retransmit timeout (rtt) is calculated using following algorithm [30]. Let α and β be the coefficients that are used for the smooth change of retransmission timeout. The $flag$ indicates that

the new retransmission time is to be calculated. The initial values are

$$rtt = INITIAL\_RTT \tag{4.1}$$

$$smooth\_rtt = \frac{rtt}{\beta} \tag{4.2}$$

Whenever the flag is set and an acknowledge arrives, the *rtt* is updated.

$$rtt1 = current\_time - base\_time, \tag{4.3}$$

where *base time* is the time that the corresponding packet was sent. Therefore the *rtt*1 is the elapsed time between the packet transmission and the acknowledgement arrival (round trip delay). The *smooth_rtt* and *rtt* are updated

$$smooth\_rtt = \alpha \times smooth\_rtt + (1 - \alpha) \times rtt1 \tag{4.4}$$

$$rtt = \beta \times smooth\_rtt \tag{4.5}$$

If the new *rtt* is greater than a maximum value or less than a minimum value, the maximum or the minimum value is used for the *rtt*. When the new *rtt* is set, the *flag* is reset to prevent another update. The *flag* is set again when the next packet is transmitted. If the *flag* is 0 when a packet is transmitted, the time of transmission is recorded and the *flag* is set to 1 to notify that a new *rtt* is ready to be updated.

Timer Design

The timers are implemented using an interrupt function. The interrupt is made to occur at the designated time. The timers implemented in the RSM-TP are as follows.

T_RETRANS Packet transmission $\longleftrightarrow$ Acknowledgement arrival

- Set when a packet is transmitted.
- Reset when the acknowledgement arrives.
- On timeout, retransmit the corresponding packet from retransmit list.

T_RECV Packet into RCL $\longleftrightarrow$ Next packet into RCL

- Set when a packet is inserted into RCL.
- Reset when the next packet is inserted into RCL.
- On timeout, the next expected sequence number is increased, and RL is searched for any eligible packet for RCL.

T_EQ_SEND First equal synchronization packet arrival from user $\longleftrightarrow$ Last equal synchronization packet arrival from user

- Set when the 1st packet to be synchronized (EQUAL) has arrived from its user.
- Reset when all packets to be synchronized (EQUAL) has arrived.
- On timeout, synchronization sequence number is increased for the next synchronization.

T_EQ_RECV First equal synchronization packet arrival to receiver $\longleftrightarrow$ Last equal synchronization packet arrival to receiver.

- Set when the 1st packet to be synchronized (EQUAL) has arrived at receiver.
- Reset when all packets to be synchronized (EQUAL) has arrived.

- On timeout, only arrived packets are sent to their users.

T_NEQ1 SYNC packet from network ⟷ Arrival of the 1st synchronization (Non-EQUAL) packet.

- Set when SYNC packet arrives from the network.
- Reset when the 1st packet to be synchronized (Non-EQUAL) has arrived.
- On timeout, NON-EQUAL synchronization error.

T_NEQ2 First synchronization packet (Non-EQUAL) ⟷ Second synchronization packet.

- Set when the 1st packet to be synchronized is sent to its user.
- Reset on time-out.
- On timeout, send 2nd Non-EQUAL packet to its user.

Transport Protocol Data Unit (TPDU) Design

The synchronization of multiple channels is done using two pieces of information. One is contained in the SYNC field of DT-TPDU, and the other is contained in the SYNC-TPDU packet which is sent through the separate SYNC channel. Those two types of information can be used either separately or in combination according to the type of synchronization.

DT-TPDU with Synchronization Information

The synchronization information in the DT-TPDU handles the simplest EQUAL synchronization: the specified packets in the multiple channels must be passed to their

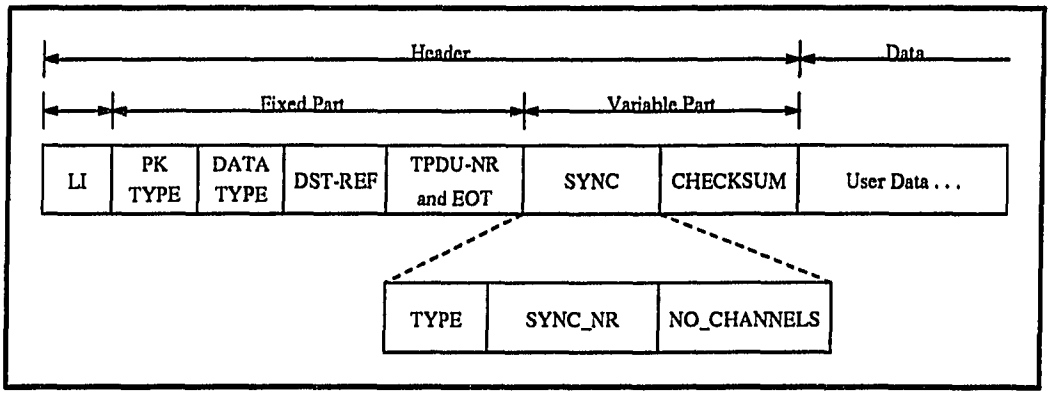

Figure 4.8: DT-TPDU with Synchronization Information .

transport protocol user within a predefined time at the remote site. In this case, the remote transport protocol waits for the packets that have the same SYNC number. The necessary synchronization information can be written in the variable header part of the DT-TPDU, and the SYNC information is analyzed when the header is decoded. The format of the DT-TPDU is defined as in Figure 4.8. The DT-TPDU of the RSM-TP is similar to that of the OSI transport protocol (OSI-TP). In the variable part of the OSI-TP header, only the CHECKSUM field is currently defined. The RSM-TP uses the variable part of the header to carry the synchronization information. That information is designed to be small so as not to increase overhead in the packet.

In the EQUAL synchronization case, only the number of channels specified in *NO_CHANNELS* field has the same synchronization sequence number in *SYNC_NR* field. The receiver will wait for the number of channels for the predefined allowable time. If the number of arrived packets is not equal to the *NO_CHANNELS* and the timer (T_EQ_RECV) expires, only the received packets are passed to the user and an error is reported. This DT-TPDU packet with slight modification can handle the simplest EQUAL synchronization with minimum overhead. Table 4.2 shows the

Table 4.2: DT-TPDU Parameters

Field Name		Length (bits)	Value
LI		8	Length of Header except LI
PK-TYPE		4	DT (1111)
DATA-TYPE		4	Voice, Video, Sdata, Idata
DST-REF		16	Destination Reference
EOT		1	1 if End of Multiple TPDU
TPDU-NR		31	Sequence Number
CHECKSUM		24	Checksum for the Entire TPDU
SYNC	SYNC TYPE	8	Not-SYNC, EQUAL, Non-EQUAL, COMB
	SYNC-NR	32	SYNC Sequence Number
	NO-CHANNELS	8	Number of Channels to be Synchronized

parameters in the DT-TPDU header with synchronization capability.

SYNC Channel

Non-Equal or more complex combined synchronization is achieved using a separate independent SYNC channel. The SYNC channel is a separate data channel in the network layer. For proper functioning of these types of synchronization, a relatively large amount of data containing the necessary synchronization information are sent through the SYNC channel using a SYNC-TPDU. The SYNC-TPDU is newly-defined packet and is to be added to the OSI-TP specification without conflict. In the OSI-TP specification, four bits in the header part are used to specify the packet type. Table 4.3 shows the code for the packet types of the OSI-TP specification. Out of 16 (2^4) available codes, ten codes are currently specified in the specification, and four codes are already in use in related protocols defined by standards organizations other than CCITT/ISO [31]. The remaining two codes are available for this SYNC-TPDU. At this design, 0100 (4) is used.

Table 4.3: OSI TPDU Code for Packet Type

TPDU	Function	Code
CR	Connection Request	1110 (E)
CC	Connection Confirm	1101 (D)
DR	Disconnect Request	1000 (8)
DC	Disconnect Confirm	1100 (C)
DT	Data	1111 (F)
ED	Expedited Data	0001 (1)
AK	Data Acknowledge	0110 (6)
EA	Expedited Data Acknowledge	0010 (2)
RJ	Reject	0101 (5)
ER	TPDU Error	0111 (7)
Not Available (Used by Other Protocols)		0000 (0)
		0011 (3)
		1001 (9)
		1010 (A)
Not Used		0100 (4)
		1011 (B)

The SYNC-TPDU is still a very small size packet compared with the normal channel data, therefore only slight amount of overhead will be added to the total network traffic. The data inside the SYNC-TPDU contain synchronization type, channels to be synchronized (CH1, CH2), sequence number of each channel, and timing parameters for proper synchronization. In this design, two data packets are to be synchronized. The data in CH1 are always delivered to the user no later than the data in CH2. Therefore the timing value is always zero or a positive value. The format of the SYNC-TPDU is shown in Figure 4.9. Table 4.4 shows the parameters inside the SYNC-TPDU.

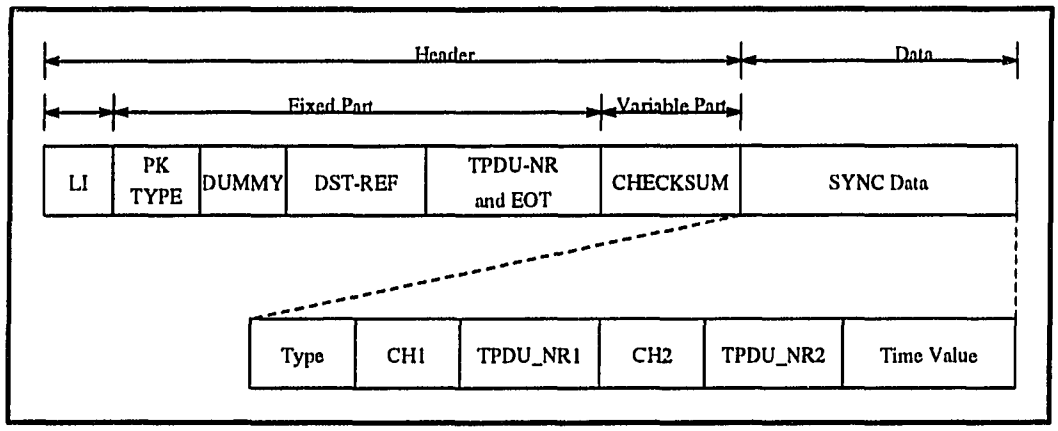

Figure 4.9: Structure of SYNC-TPDU

Table 4.4: SYNC-TPDU Parameters

Field Name		Length (bits)	Value
Header	LI	8	Length of Header except LI
	PK-Type	4	SYNC (0100)
	Dummy	4	Not used (0000)
	DST-REF	16	Destination Reference
	EOT	1	1 if End of Multiple TPDU
	SYNC-TPDU-NR	31	Sequence Number
	CHECKSUM	24	Checksum for the Entire TPDU
Data	SYNC TYPE	8	Not-SYNC, EQUAL, Not-EQUAL, COMB
	CH1	8	First Channel
	TPDU-NR1	32	Sequence Number of CH1
	CH2	8	Second Channel
	TPDU-NR2	32	Sequence Number of CH2
	Time Value	32	Delay between CH1 and CH2

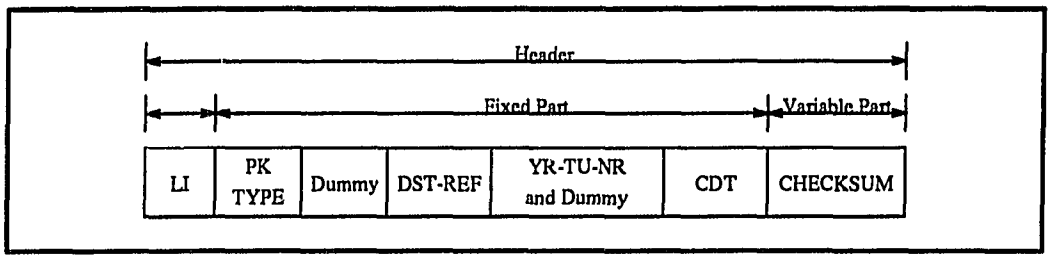

Figure 4.10: Structure of AK-TPDU

Table 4.5: AK-TPDU Parameters

Field Name	Length (bits)	Value
LI	8	Length of Header except LI
PK-TYPE	4	AK (0110)
Dummy	4	Not Used (0000)
DST-REF	16	Destination Reference
YR-TU-NR	31	Next Expected DT-TPDU Number
Dummy	1	Not Used (0)
CDT	16	Credit
CHECKSUM	24	Checksum

Acknowledgement (AK)-TPDU

The AK-TPDU is the same packet as defined in the OSI-TP. The extended format is used for higher throughput. In the extended format, 31 bits are used for the sequence number of the next expected DT-TPDU, and 16 bit credit (CDT) is used. The format of the AK-TPDU is shown in Figure 4.10, and Table 4.5 shows its parameters.

CHAPTER 5. MODELING OF RSM-TP

Based on the design of the RSM-TP, a simulation queueing model is defined and analyzed. The queueing model is simplified and analyzed, and simulation results are also given to validate the model. In the simplified model, the arrivals are Poisson processes, all queues are made to have exponential service time distribution, and the propagation delay is set to zero. With the above simplification, we can analyze all the queues as M/M/1 queueing models. Figure 5.1 shows the queueing model of the RSM-TP network for one-side communication.

Specifications

The RSM-TP is to be designed so that it is close to the real network environment, also it is easily modeled and analyzed.

Data Channels

Five channels are made for the RSM-TP network. The three channels (Voice, Video, Sdata) are synchronized channels, Idata is an independent data channel, and SYNC is a channel for synchronization information. For EQUAL synchronization, the number of the synchronized channels is from 0 to 3, i.e. any combination of the three channels can be synchronized. For Non-EQUAL synchronization, the SYNC

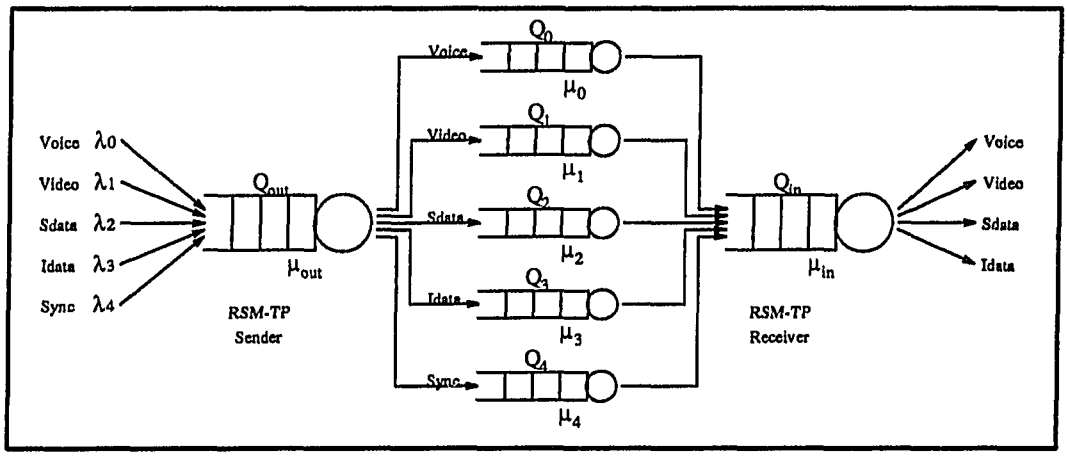

Figure 5.1: Queueing Model of RSM-TP

Table 5.1: Average Network Traffic Parameters

	Packet Size (bits)	Inter-arrival Time (secs)	Data Rate (bits/sec)	Data Rate with Header or ACK (bits/sec)
Voice	128	2.000×10^{-3}	64 K	132.000 K
Video	8192	8.192×10^{-5}	100 M	101.660 M
Sdata	1024	1.024×10^{-3}	1 M	1.133 M
Idata	8192	8.192×10^{-4}	10 M	10.290 M
SYNC	120	1.500×10^{-2}	8 K	13.867 K

packet can point any of the two DT-TPDUs in the same or different channels.

Average Data Rate and Network Traffic

The transport layer service data unit (TSDU) is sent to the underlying network with the header of TPDU. The header size of a DT-TPDU is 136 bits, and the header size of a SYNC-TPDU is 88 bits. The average packet size and inter-arrival time are determined to make the average data rate in Table 5.1

Table 5.2: Network Service Rate in Bits

	Simplified Model (bps)	Real Model (bps)
Q_0 (Voice)	64 K × 1.1 = 70.40 K	132 K × 1.1 = 145.2000 K
Q_1 (Video)	100 M × 1.1 = 110.00 M	101.66 M × 1.1 = 111.8260 M
Q_2 (Sdata)	1 M × 1.1 = = 1.10 M	1.133 M × 1.2 = 1.3596 M
Q_3 (Idata)	10 M × 1.1 = 11.00 M	10.29M × 1.2 = 12.3480 M
Q_4 (Sync)		13.867 K × 1.2 = 16.64 K (10 M)
Q_{in} Q_{out}	122.17 M	125.6954 M

If acknowledge is enabled, the ACK packet is sent for the each DT/SYNC-TPDU. The size of the SYNC-TPDU is 208 bits (88 bits of header + 120 bits of data). The required average network data rate is more than the average pure data rate because of the overhead of header part of each TPDU. The ACK-TPDU also consumes some of the network bandwidth. The data rate of real-time data with acknowledge are also shown, where header size is added and acknowledgement is used for Idata channel.

Network Service Rate

According to the data rate and the network traffic, the service rate of each network queue is given in Table 5.2. The service rate of the Sync channel (μ_4) without header (simplified model) is not considered because the rate without header is only for simplification of analysis. In that case, only four data channels are set up for normal data transmission. In the simplified analysis, all network data rates are assumed to be 1.1 times the network traffic. In more realistic model, Voice and

Video data streams are considered as constant streams of packets. Their inter-arrival time and packet size have constant distribution. The synchronized data (Sdata) is considered to have uniformly distributed inter-arrival time and constant packet size. The independent data (Idata) is the normal data traffic, and its inter-arrival time and packet size are made to have exponential distributions as in the ideal queue case. The network capacity is assumed to be 1.1 times the total traffic for the Voice and Video channel. The Sdata and Idata channels are given 1.2 times the traffic, since their non-constant distributions result in relatively long unstable state during simulation. Also, acknowledgement is applied for Idata channel. The Q_4 for the SYNC is given 10 Mbps service rate, which is close to the independent data channel. Although the Sync channel has small amount of traffic, it has to arrive to the destination as soon as possible to minimize the synchronization error.

Queueing Model Analysis

The queueing model in Figure 5.1 is a simplified queueing network. With exponentially distributed inter-arrival time and service time, the model is analyzed as M/M/1 queueing networks. The delay in the queue and the size of the queue are calculated. The packet size is used without TPDU header to preserve the exponential service time distribution.

Transmit/Receive Queue (Q_{out}, Q_{in})

The Q_{out} and Q_{in} have the same arrival processes and the same service time distributions for the simplified queueing model. The analysis for the Q_{out} can be applied to the Q_{in}. The arrival rate is $1/interarrival\ time$. The arrival of Q_{out}

Table 5.3: Queue Parameters

	λ	$\overline{m}$	μ
Q_{out}	14905	7452	16394
Q_0	500	128	550
Q_1	12207	8192	13428
Q_2	977	1024	1074
Q_3	1221	8192	1343
Q_{in}	14905	7452	16394

consists of the arrivals from four independent sources, where the inter-arrival time between packets for each source is exponentially distributed (Poisson process) with parameter λ_i, where $i = 0, 1, 2, 3$. The merged stream is also a Poisson process with parameter λ, where

$$\lambda = \lambda_0 + \lambda_1 + \lambda_2 + \lambda_3 \tag{5.1}$$

The average packet size ($\overline{m}$) of the arrival in Q_{out} is given

$$\overline{m} = \frac{1}{\lambda} \times \sum_{i=0}^{3} (\overline{m_i} \times \lambda_i), \tag{5.2}$$

where $\overline{m_i}$ is the average packet size of channel i. The packet service rate (μ_{out}) is given

$$\mu_{out} = \frac{service\ rate\ in\ bits}{average\ packet\ size(\overline{m})} \tag{5.3}$$

The arrival rate, the average message size, and the service rate is summarized in Table 5.3. The utilization (ρ_{out}) of the Q_{out} is

$$\rho_{out} = \frac{\lambda}{\mu_{out}} \tag{5.4}$$

The average number of packets ($\overline{n}_{out}$) in the queue is

$$\overline{n}_{out} = \frac{\rho_{out}}{1 - \rho_{out}} \tag{5.5}$$

The average delay ($\bar{d}_{out}$) of the packet in the Q_{out} is the sum of the average queueing time and the average service time of the packet, and can be calculated using Little's law.

$$\bar{d}_{out} = \frac{\bar{n}_{out}}{\lambda} \tag{5.6}$$

The delay $\bar{d}_{out}$ is the average delay of all packets in the Q_{out}. The average delay of each data type can be calculated. The different type of data has different mean delay in the Q_{out}. To calculate the mean delay of each data type, the queue is to be considered as an M/G/1 queue. Let $m_0(t)$, $m_1(t)$, $m_2(t)$, and $m_3(t)$ be the probability density function (pdf) of the packet size (transmission time) of each data type. For an M/G/1 queue, the probability generating function of the number of packets in the queue ($P(z)$) is

$$P(z) = E[z^n] = \frac{(1-\rho)(1-z)A(z)}{A(z)-z}, \tag{5.7}$$

where $A(z)$ is the probability generating function of Poisson arrivals during the random packet transmission time. Since each type of packet arrival is a Poisson process and independent, the probability of type i packet transmission is

$$\frac{\lambda_i}{\lambda 0 + \lambda_1 + \lambda_2 + \lambda_3} \equiv \frac{\lambda_i}{\lambda} \tag{5.8}$$

Let T_i be the transmission time of data type i.

$$A(z) = E[z^a] \tag{5.9}$$

$$= \sum_{i=0}^{3} \left\{ \frac{\lambda_i}{\lambda} \int_0^\infty E\left[z^a \middle| T_i = t\right] m_i(t)dt \right\} \tag{5.10}$$

$$= \sum_{i=0}^{3} \left\{ \frac{\lambda_i}{\lambda} \int_0^\infty \sum_{n=0}^{\infty} z^n \frac{(\lambda t)^n e^{-\lambda t}}{n!} m_i(t)dt \right\} \tag{5.11}$$

$$= \sum_{i=0}^{3} \left\{ \frac{\lambda_i}{\lambda} \int_0^{\infty} e^{-\lambda t(1-z)} m_i(t) dt \right\} \tag{5.12}$$

$$= \sum_{i=0}^{3} \frac{\lambda_i}{\lambda} M_i \left[\lambda(1-z)\right] \tag{5.13}$$

$$\equiv M \left[\lambda(1-z)\right] \tag{5.14}$$

With the same arguments

$$P(z) = E\left[z^n\right] \tag{5.15}$$

$$= \sum_{i=0}^{3} \left\{ \frac{\lambda_i}{\lambda} \int_0^{\infty} E\left[z^n | delay = t\right] d_i(t) dt \right\} \tag{5.16}$$

$$= \sum_{i=0}^{3} \frac{\lambda_i}{\lambda} D_i \left[\lambda(1-z)\right], \tag{5.17}$$

where $d_i(t)$ is the pdf of the delay of data type i.

By combining the equations (5.7), (5.14), and (5.17)

$$P(z) = \frac{(1-\rho)(1-z) \sum_{i=0}^{3} \frac{\lambda_i}{\lambda} M_i \left[\lambda(1-z)\right]}{M\left[\lambda(1-z)\right] - z} = \sum_{i=0}^{3} \frac{\lambda_i}{\lambda} D_i \left[\lambda(1-z)\right] \tag{5.18}$$

The delay of each data type is

$$D_i \left[\lambda(1-z)\right] = \frac{(1-\rho)(1-z) M_i \left[\lambda(1-z)\right]}{M\left[\lambda(1-z)\right] - z} \tag{5.19}$$

By substituting s for $\lambda(1-z)$ and $1 - s/\lambda$ for z,

$$D_i(s) = \frac{s(1-\rho)}{s - \lambda + \lambda M(s)} \times M_i(s) = D_q(s) \times M_i(s), \tag{5.20}$$

where $D_i(s)$, $D_q(s)$ and $M_i(s)$ are Laplace transformations of the delay, queueing time, and service time pdf. The mean delay can be obtained by differentiation of the $D_i(s)$.

$$\overline{d_i} = \overline{d_q} + \overline{m_i} \tag{5.21}$$

The equation (5.21) says that any arriving packet has the same queueing time in the queue regradless of the data type. The mean queueing time ($\overline{dq_{out}}$) for the M/M/1 queue is

$$\overline{dq_{out}} = \frac{\rho_{out}/\mu_{out}}{1-\rho_{out}} \tag{5.22}$$

The mean delay of each data type can be calculated by adding the mean service time to the mean queueing time.

$$\overline{d_i}_{out} = \frac{\overline{m_i}}{\mu_{bout}} + \overline{dq_{out}}, \tag{5.23}$$

where μ_{bout} is the service rate of Q_{out} in bits.

Network queues

A Poisson process driving an exponential server generates a Poisson process for departure (Burke's Theorem) [32], that is, the inter-departure time distribution is exponential, and that becomes the inter-arrival time distribution of the next queue. The departure process of Q_{out} is divided into each transmit queue with probability λ_i/λ. The split process is also a Poisson process with parameter

$$\lambda \times \frac{\lambda_i}{\lambda} = \lambda_i \tag{5.24}$$

Therefore the network queues can be analyzed as separate M/M/1 queues. But that is true only under the assumption that the servers are independent processes, and the arrival process and the service process are independent. In communication network it is not always possible since the service time depends upon the length of the message which is carried from queue to queue [33]. This introduces dependency between the arrival process and the length of message (service time). For the purpose of analysis, each queue is assumed to behave as an independent server to the other.

CHAPTER 6. SIMULATION

The basic network queues in this design are measured with ideal parameters and compared with the calculated results to validate the basic queueing model. The simulation of the designed transport protocol is performed for the different synchronization types with more realistic network parameters. The end-to-end delays and the receive buffers used for synchronization are measured during simulation. All arrivals are not Poisson processes, and also the departure process is affected by the service time in the queue. The propagation delay could be given to the network, but since this can be regarded as a constant that can be adjusted, the simulation is performed with zero propagation delay. The acknowledgement packets are generated for the independent data channel (Idata). The network error rate is given as a network parameter (errors/bit).

Basic Queueing Model

The simplified in/out queues and network queues are measured with exponential inter-arrival time and exponential service time distribution. The average packet sizes and average delays in the network queues are measured. Figure 6.1 shows the average number of packets and the average delay in the I/O queues. Figure 6.2 shows the average delay of I/O queues for each data type. Figure 6.3 shows the average number

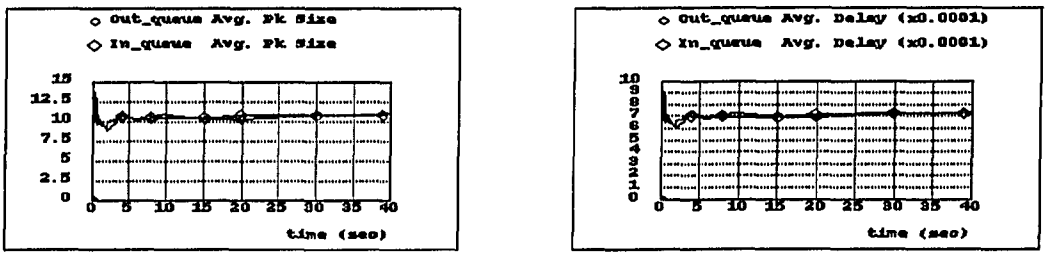

Figure 6.1: Average Packet Size and Delay in I/O Queues

of packets and the average delay in the network queues. Figure 6.4 shows the average end-to-end delay of each channel. Table 6.1 and Table 6.2 shows the analytical results and the simulation results for the ideal queue assumptions. In Q_{out} and Q_{in}, **Total** is the average time spent in the queue by any data type, and **Queueing** is the average queueing time in the queue, which is the same for all data types. The **End-to-End** delay is the sum of the delays in Q_{in}, Q_{out}, and network queue. During simulation, the packet coming from the user is time-stamped, and the elapsed time is measured when the packet is sent to its user at the destination. Since all queues are assumed to be M/M/1 queues and the service rate is given 1.1 times of the data rate, the number of packets in the queue is same for all queues (10.0132 packets). The simulations are performed for 80 seconds except for the packet size and delay for I/O queues, which have relatively early stablization.

RSM-TP Simulation Parameters

The synchronization operation of the RSM-TP is simulated with non-ideal parameters. The inter-arrival times of Voice and Video have constant, Sdata and SYNC have uniform, and Idata has exponential distribution respectively. For the packet size, Voice, Video, Sdata and SYNC have constant, and Idata has exponential distribution

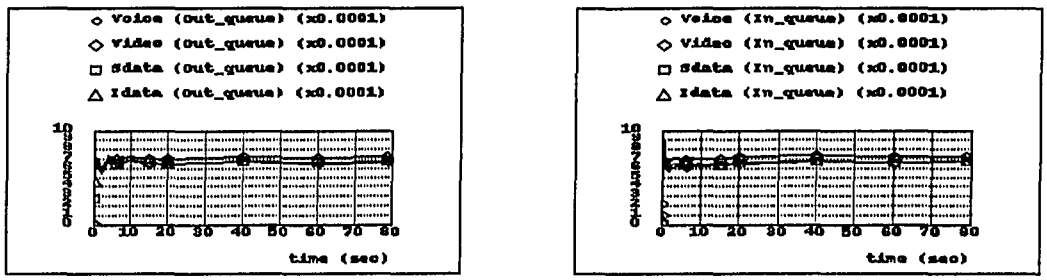

Figure 6.2: Average Delay of Each Data in I/O Queues

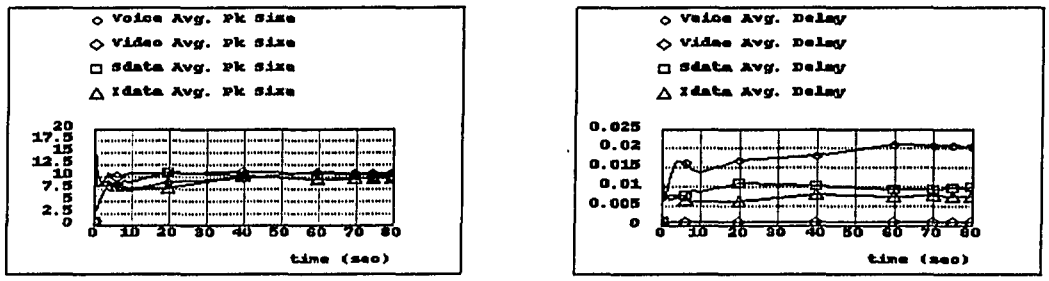

Figure 6.3: Average Packet Size and Delay in Network Queues

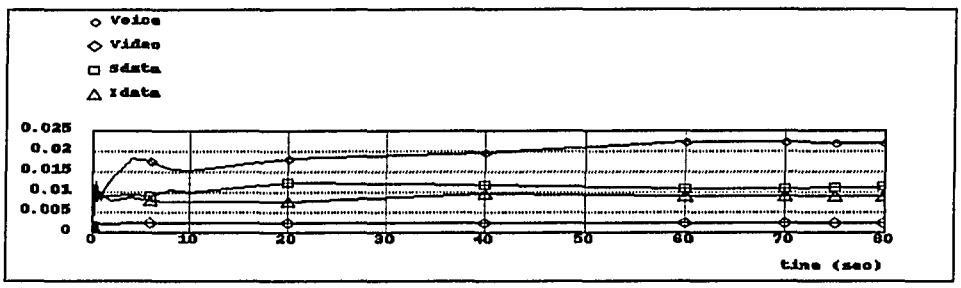

Figure 6.4: Average End-to-End Delay

Table 6.1: Simulation Comparison of Mean Packet Size

	Analytical	Simulation	Difference (%)
Q_{out}	10.0132	10.75	7.36
Q_0	10.0132	10.75	7.36
Q_1	10.0132	10.58	5.66
Q_2	10.0132	10.03	0.17
Q_3	10.0132	9.70	-3.13
Q_{in}	10.0132	10.89	8.76

Table 6.2: Simulation Comparison of Mean Delay

		Analytical (secs)	Simulation (secs)	Difference (%)
Q_{out}	Total	6.7180×10^{-4}	7.22×10^{-4}	7.47
	Queueing	6.1079×10^{-4}	n/a	n/a
	Voice	6.1184×10^{-4}	6.61×10^{-4}	8.03
	Video	6.7784×10^{-4}	7.22×10^{-4}	6.51
	Sdata	6.1917×10^{-4}	6.66×10^{-4}	7.56
	Idata	6.7784×10^{-4}	7.23×10^{-4}	6.66
Network	Q_0 (Voice)	2.0000×10^{-2}	2.05×10^{-2}	2.50
	Q_1 (Video)	8.1920×10^{-4}	8.25×10^{-4}	0.71
	Q_2 (Sdata)	1.0235×10^{-2}	9.85×10^{-3}	-3.76
	Q_3 (Idata)	8.1900×10^{-3}	7.50×10^{-3}	-8.42
Q_{in}	Total	6.7180×10^{-4}	7.31×10^{-3}	8.81
	Queueing	6.1079×10^{-4}	n/a	n/a
	Voice	6.1184×10^{-4}	6.70×10^{-4}	9.51
	Video	6.7784×10^{-4}	7.25×10^{-4}	6.96
	Sdata	6.1917×10^{-4}	6.70×10^{-4}	8.21
	Idata	6.7784×10^{-4}	7.25×10^{-4}	6.96
End-to-End	Voice	2.1223×10^{-2}	2.18×10^{-2}	2.72
	Video	2.1749×10^{-3}	2.28×10^{-3}	4.83
	Sdata	1.1473×10^{-2}	1.12×10^{-2}	-2.38
	Idata	9.5457×10^{-3}	8.95×10^{-3}	-6.24

Table 6.3: Simulation Parameters

Parameters		Values	Description
T		10 secs	Simulation Time
Bit Error Rate		1×10^{-6}	Network Bit Error Rate
Err Threshold	VOICE	0.10	Acceptable Bit Error Rate
	VIDEO	0.01	
	SDATA	0.00	
	IDATA	0.00	
	SYNC	0.00	
Prop. Delay		0.0 secs	Network Propagation Delay
T_RETRANS	MIN	0.05 secs	Minimum Retransmit Time
	MAX	0.5 secs	Maximum Retransmit Time
	INIT	0.1 sec	Initial Retransmit Time
T_RECV		0.005 sec	Timeout Between Packet Arrivals
T_EQ_SEND		0.005 sec	Timeout for EQUAL Sync. at Sender
T_EQ_RECV		0.005 sec	Timeout for EQUAL Sync. at Receiver
T_NEQ1		0.005 sec	Timeout for 1st Non-Equal Packet Arrival
T_NEQ2		0.005 sec	Timeout for 2nd Non-Equal Packet Arrival

respectively. Their detailed parameters were shown in Table 4.1. Other parameters for this simulations are shown in Table 6.3. The network error rate is assumed to be 10^{-6} errors per bit. The bit error threshold of the Voice and Video are relatively large due to their characteristics. The Sdata, Idata, and SYNC allows no bit error. A packet is accepted as a valid one if the number of bit errors in the packet is less than *Err Threshold* $\times$ *Packet Size*. The invalid packet is considered to be a lost packet. For the real-time data (Voice, Video, and Sdata), the lost packets result in increase of end-to-end delay of other synchronized channels, i.e., the other channels wait for the time-out of the packet on the channel. For the Idata, a lost packet does not generate an acknowledge packet, which makes the sender retransmit the packet increasing both retransmit queue size and receive queue size. T_RETRANS is used

for Idata channel, which is not related to synchronization. All timer parameters related to synchronization are given a value of 0.005 seconds.

End-to-End Delay

The end-to-end delay is the time taken by the transport layer and its subnetworks. When a packet arrives from its user, the packet is time-stamped. When the packet is delivered to its destination user, its end-to-end delay is calculated. In real-time communication, not only the average delay, but also the maximum delay must be considered. If an end-to-end delay of real-time data is greater than a certain value, it means loss of information. The end-to-end delays of the designed RSM-TP are measured for several types of synchronization cases. The types of synchronization are mentioned as the following convention.

- **NS** represents no-synchronization.
- **E** represents EQUAL synchronization which performs for the three real-time channels (Voice, Video, and Sdata).
- **NE***ab* represents Non-EQUAL synchronization in which the packet in channel *a* is followed by the packet in channel *b* with specified time delay (T_NEQ2).
- COMB*ab* represents COMBINED synchronization in which all the three channels (Voice, Video, and Sdata) are EQUAL synchronized, and the channel *a* and the channel *b* perform Non-EQUAL synchronization as in the NE*ab* case. In COMB*xa*, *x* means any one of the three channels.

Figure 6.5 shows the simulation results of the end-to-end delay for no-synchronization (NS) case and Figure 6.6 shows EQUAL synchronization (E) case. The end-to-end

delay of IDATA channel is shown only at NS and EQUAL case for reference. Because the IDATA channel behaves in quite the same way for any type of synchronization and it does not participate in synchronization process directly, it is not shown at other types of synchronization. Figure 6.7 through Figure 6.15 show Non-EQUAL (NE) synchronization case, and Figure 6.16 through Figure 6.18 show combined synchronization (COMB) case.

The EQUAL synchronization slightly increases the average delay of Voice and Sdata and makes the average delay of Video almost twice as large. The more frequently arrived Video packets generate the largest number of packets in the queue waiting other channels for synchronization. The abrupt peak value of Voice and Video channel is due to an error (lost packet) of Sdata channel.

For the NE case, the channels not related to the synchronization are not affected, and remain the same as the NS case. The channel that follows the Video channel introduces small delay increasement. For the Video channel, the waiting time of 1st NE packet is relatively smaller than the packets of the other channels.

For the combined synchronization, the result is affected only by the 2nd channel, because the packets in the 1st channel wait for the packets of the other channels for EQUAL synchronization and are sent to their users at the same time, at which the timer T_NEQ2 is started. Therefore, only three different results are generated for COMB synchronization. Also, the end-to-end simulation result figures show the delay at every time instant and the average values during that time interval. The average values are recalculated every time the packet is delivered to the destination user. The measured mean delay and the maximum delay during the simulation time are summarized in Table 6.4. With the given network error rate (1×10^{-6} errors/bit),

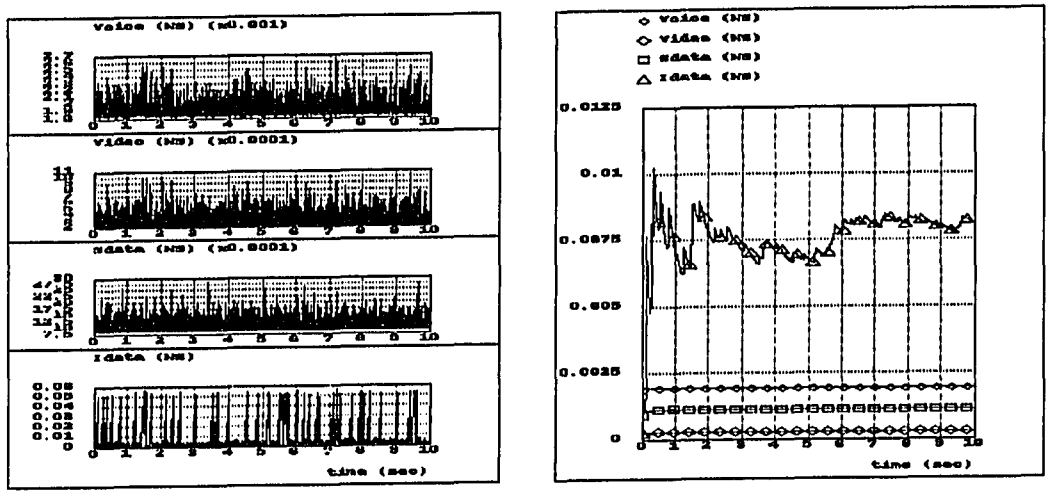

Figure 6.5: End-to-End Delay for NS

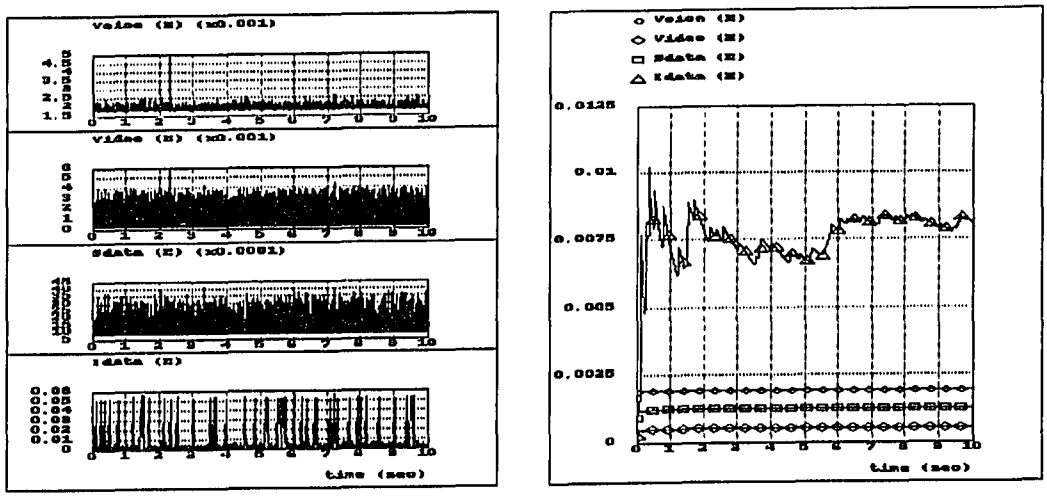

Figure 6.6: End-to-End Delay for EQUAL

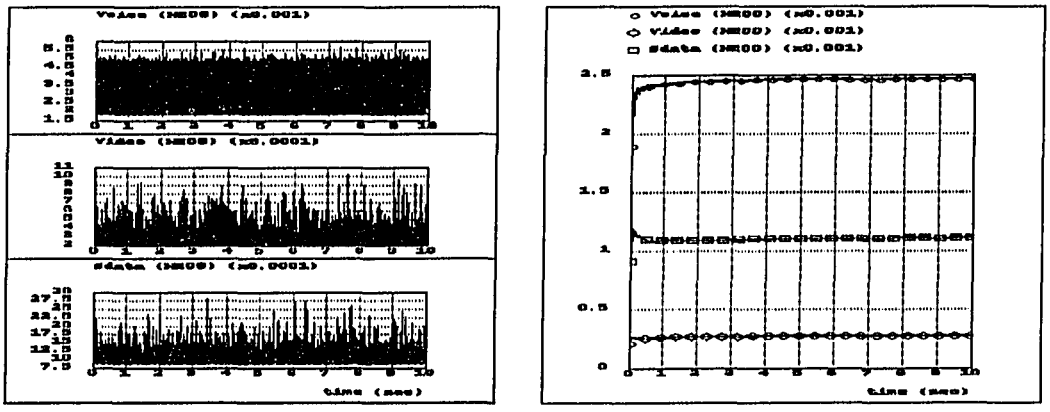

Figure 6.7: End-to-End Delay for NE00

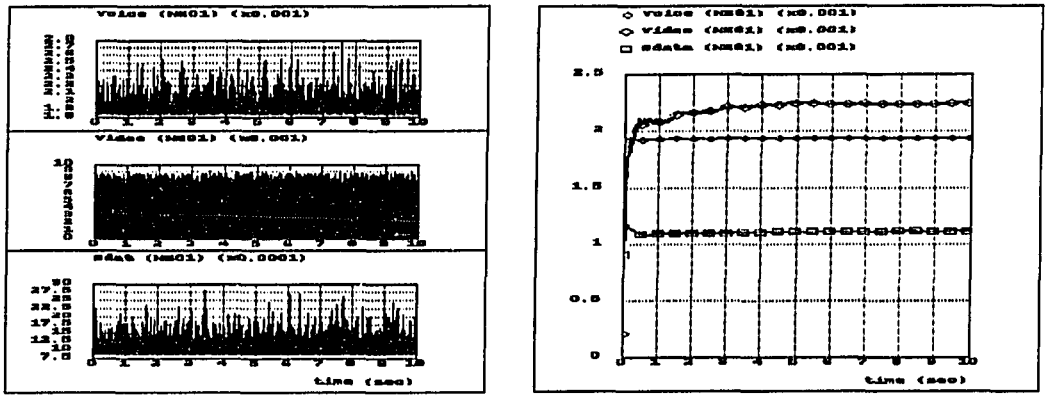

Figure 6.8: End-to-End Delay for NE01

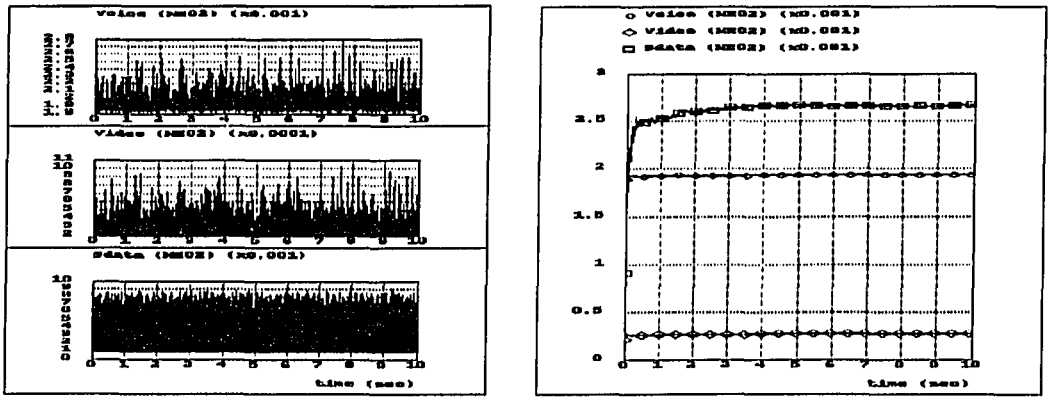

Figure 6.9: End-to-End Delay for NE02

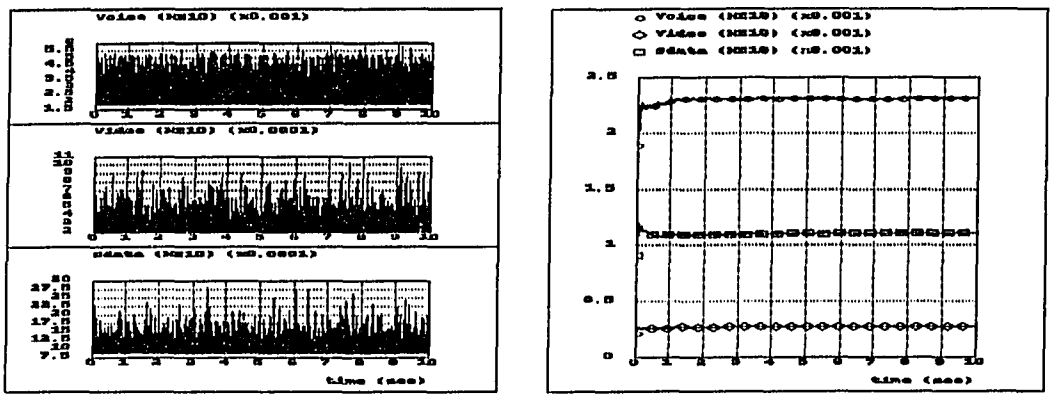

Figure 6.10: End-to-End Delay for NE10

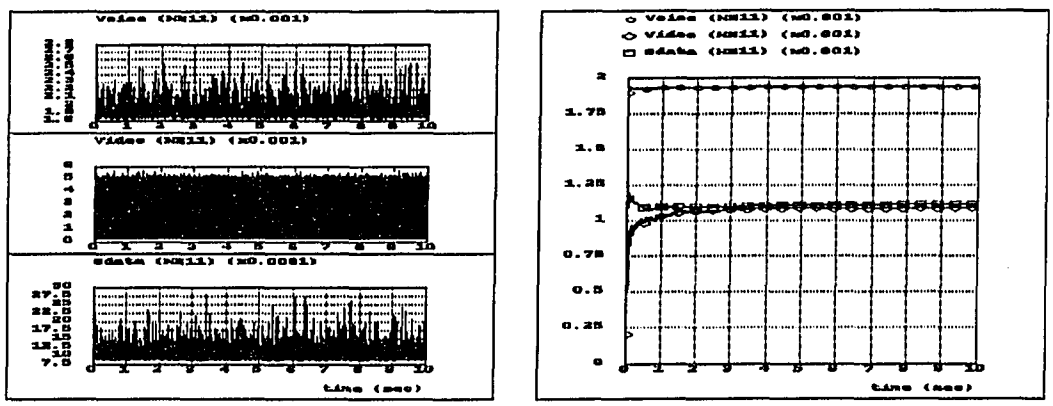

Figure 6.11: End-to-End Delay for NE11

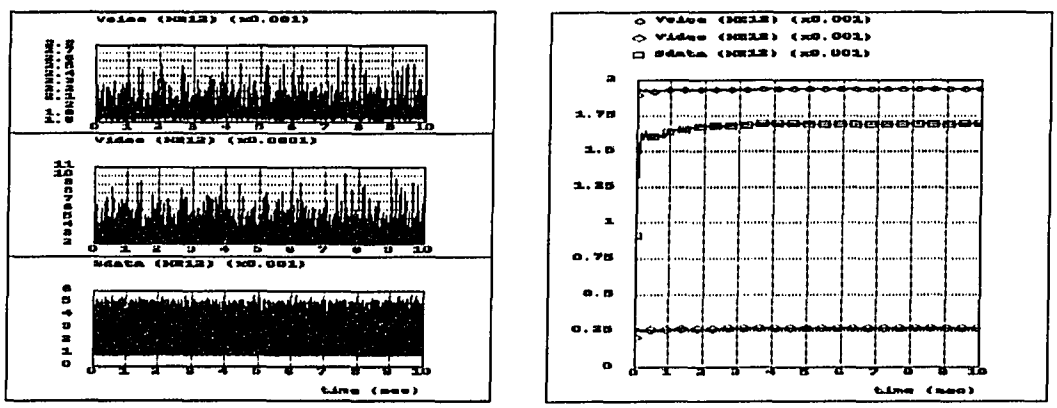

Figure 6.12: End-to-End Delay for NE12

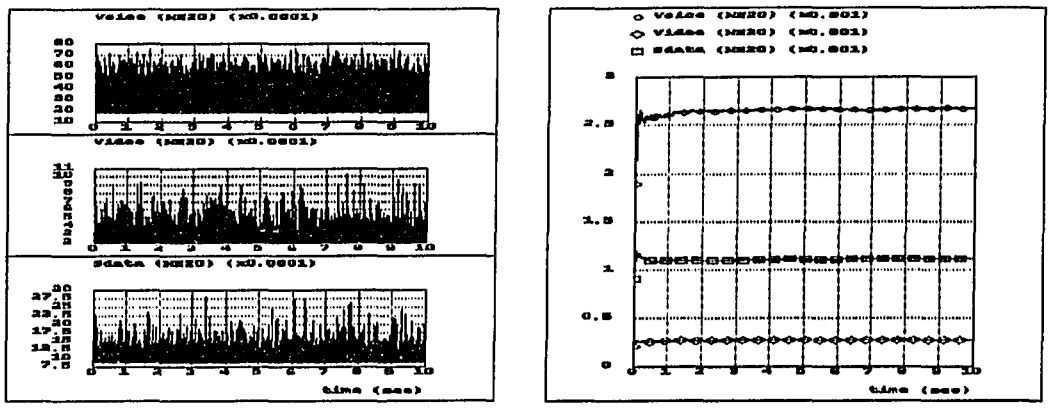

Figure 6.13: End-to-End Delay for NE20

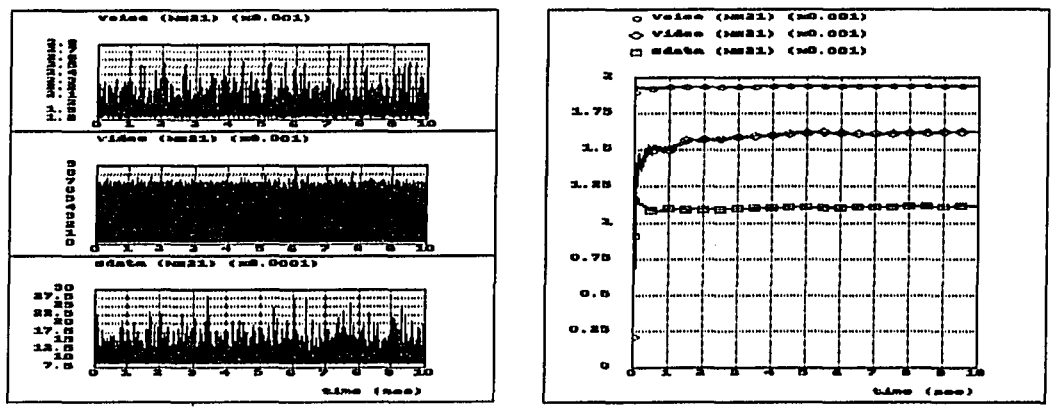

Figure 6.14: End-to-End Delay for NE21

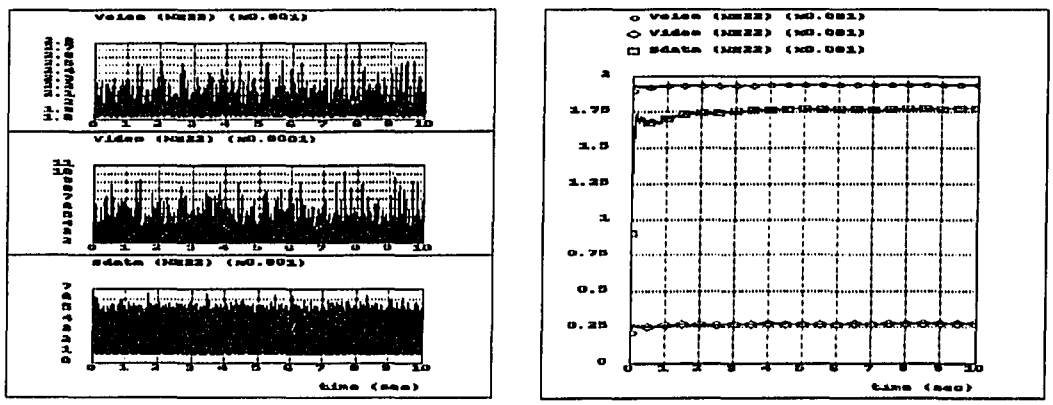

Figure 6.15: End-to-End Delay for NE22

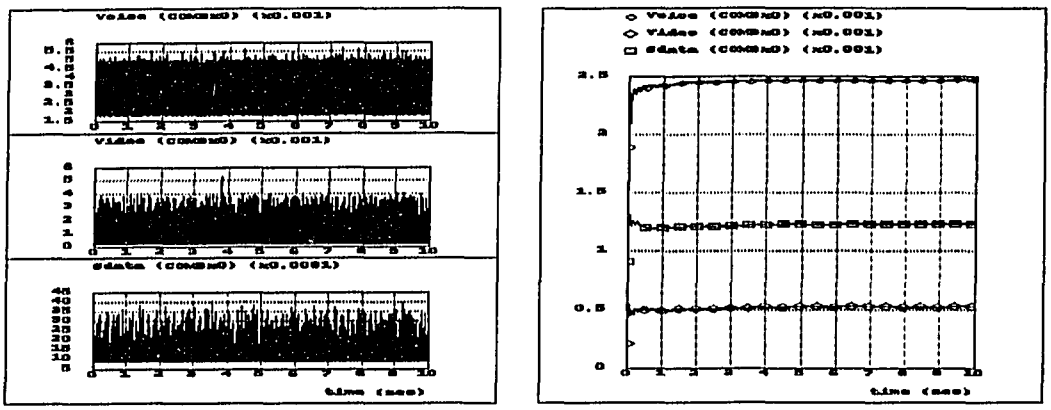

Figure 6.16: End-to-End Delay for COMBx0

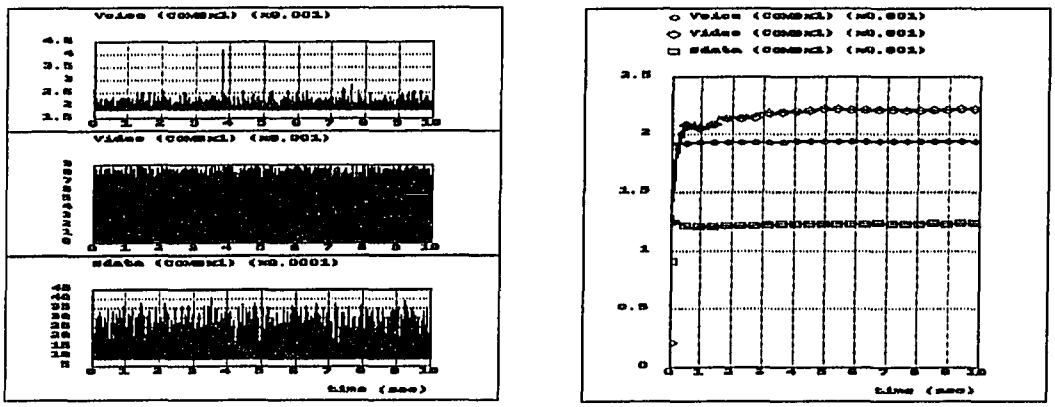

Figure 6.17: End-to-End Delay for COMBx1

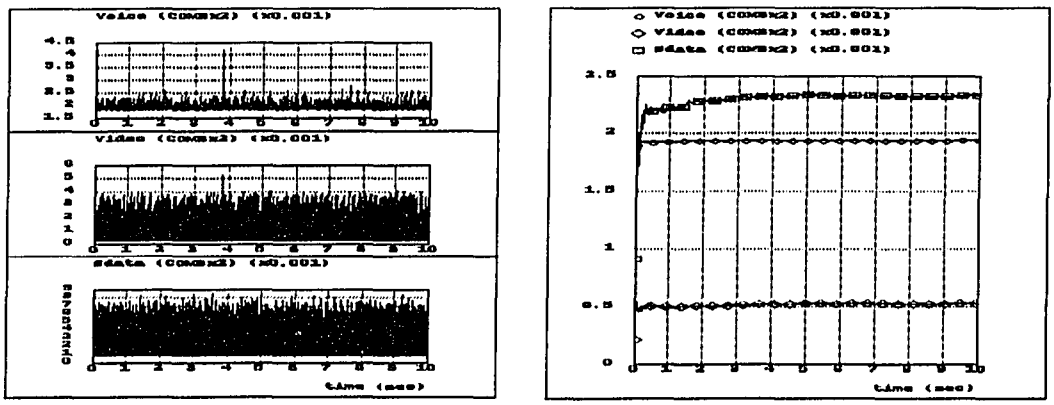

Figure 6.18: End-to-End Delay for COMBx2

Table 6.4: End-to-End Delay

	Voice ($10^{-3}secs$)		Video ($10^{-3}secs$)		Sdata ($10^{-3}secs$)		Idata ($10^{-3}secs$)	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max
$NoSync$	1.94	2.65	0.27	1.03	1.11	2.84	8.10	56.80
$EQUAL$	1.94	4.89	0.52	5.75	1.24	4.13	8.10	56.80
NE_{00}	2.47	5.64	0.27	1.02	1.11	2.80	7.73	56.25
NE_{01}	1.94	2.77	2.24	9.06	1.11	2.80	7.73	56.23
NE_{02}	1.94	2.77	0.27	1.02	2.67	9.03	7.73	56.25
NE_{10}	2.31	5.74	0.27	1.02	1.11	2.80	7.73	56.22
NE_{11}	1.94	2.77	1.09	5.72	1.11	2.80	7.73	56.22
NE_{12}	1.94	2.77	0.27	1.02	1.69	5.63	7.73	56.22
NE_{20}	2.67	7.48	0.27	1.02	1.11	2.80	7.73	56.22
NE_{21}	1.94	2.77	1.62	8.32	1.11	2.80	7.73	56.22
NE_{22}	1.94	2.77	0.27	1.02	1.77	6.53	7.73	56.22
$COMB_{00,10,20}$	2.47	5.64	0.52	5.27	1.24	4.02	7.73	56.22
$COMB_{01,21,21}$	1.94	4.19	2.21	8.98	1.24	4.02	7.73	56.22
$COMB_{02,12,22}$	1.94	4.19	0.52	5.27	2.33	8.50	7.73	56.22

only real-time data (Sdata) generates 9.22×10^{-4} measured packet error rate, and only one packet among the lost packets generates an error for EQUAL synchronization during 10 second simulation time. For the COMBab case, the delay of channels except b shows the same value as in EQUAL case. The channel b results in extra waiting time for NE synchronization, which is close value to NE0b case.

Buffer Size

The buffer affected by the synchronization function is the RCL (Receiver Contiguous List). For EQUAL synchronization, the packets wait for other packets with the same synchronization number, and for Non-EQUAL case, the packet waits for the timeout of the given timer (T_NEQ2) in the RCL. Since the packets remain in the

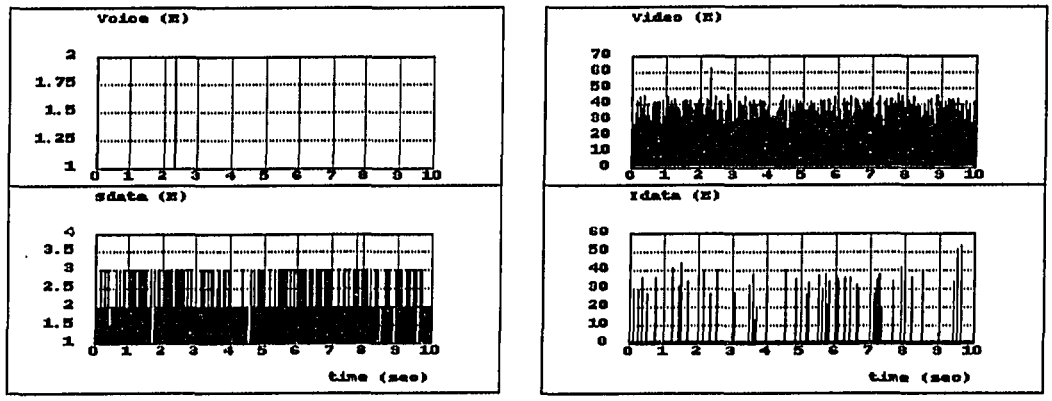

Figure 6.19: RCL Size for EQUAL

buffer only during the synchronization process, the buffer becomes empty as soon as the synchronization function is finished. Therefore, the size changes to zero abruptly and remains empty until the next synchronization cycle starts. In that sense, the average buffer size gives less meaningful information than the maximum buffer size which is the necessary buffer size for proper synchronization process. Since the buffer sizes are measured when the packets are inserted into the buffer, the minimum size is always measured to be one. The Figures 6.19 through Figure 6.21 show the receive contiguous list size for each synchronization type, and the measured average and maximum values are summarized in the Table 6.5.

For EQUAL, the voice packet has relatively constant and infrequent arrival pattern, which results in very small buffer size, usually one. The buffer size increases when there are synchronization errors. The fast arriving video packets need relatively large buffer size for synchronization. It also shows an abrupt increase in case of the synchronization error. The Sdata packets result in maximum 3-4 buffer size. For NEab case, only channel b makes the buffer grow. As in the end-to-end delay

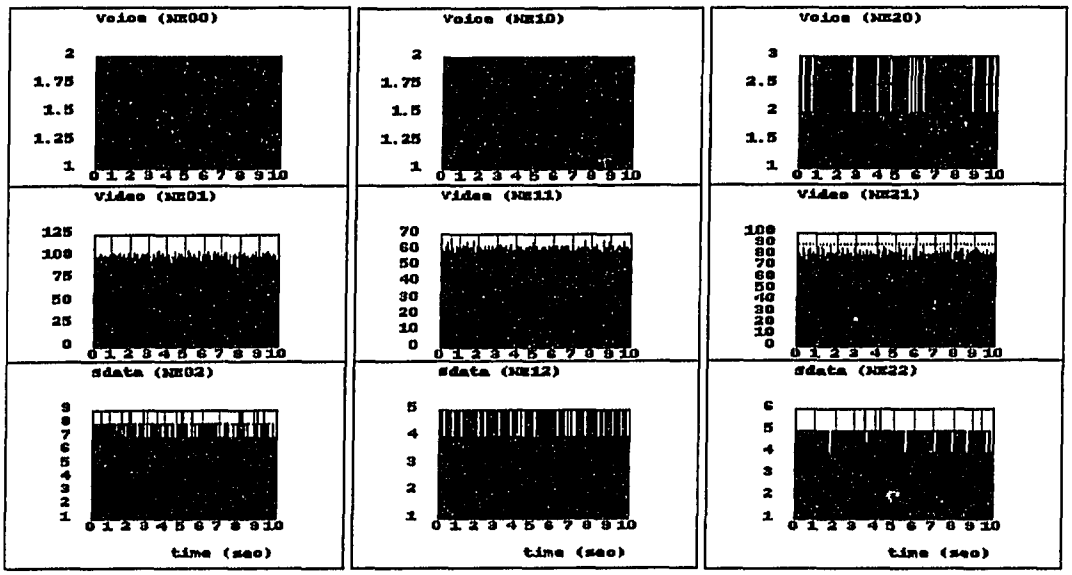

Figure 6.20: RCL Size for Non-EQUAL

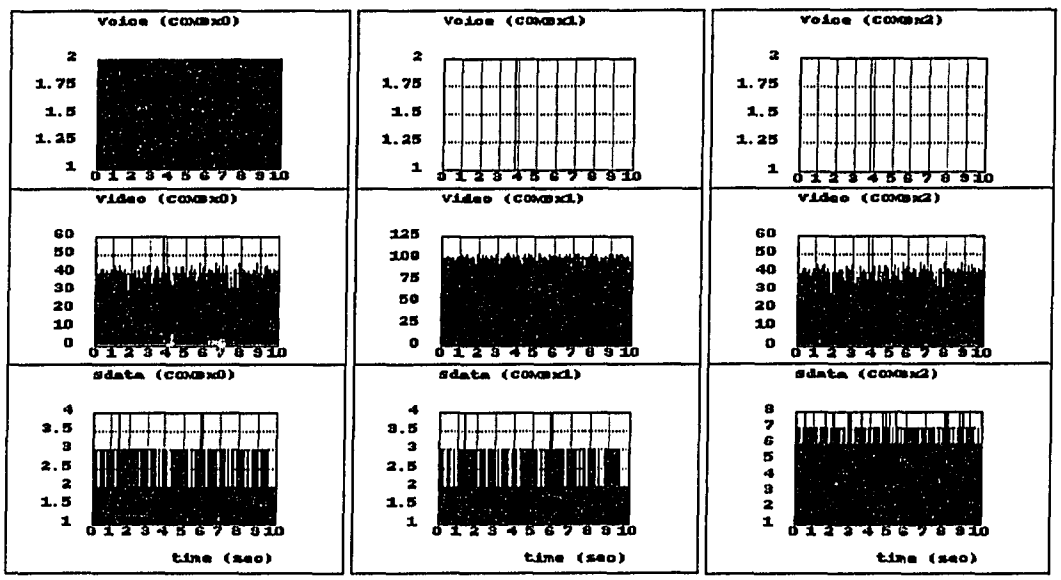

Figure 6.21: RCL Size for COMB

Table 6.5: Receive Contiguous List Size

	Voice ($10^{-3}secs$)		Video ($10^{-3}secs$)		Sdata ($10^{-3}secs$)		Idata ($10^{-3}secs$)	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max
$NoSync$	1.00	1	1.00	1	1.00	1	4.84	53
$EQUAL$	1.00	2	4.05	62	1.07	4	4.84	53
NE_{00}	1.13	2	1.00	1	1.00	1	4.75	47
NE_{01}	1.00	1	24.82	106	1.00	1	4.75	47
NE_{02}	1.00	1	1.00	1	2.31	9	4.75	47
NE_{10}	1.09	2	1.00	1	1.00	1	4.75	47
NE_{11}	1.00	1	10.80	67	1.00	1	4.75	47
NE_{12}	1.00	1	1.00	1	1.43	5	4.75	47
NE_{20}	1.22	3	1.00	1	1.00	1	4.75	47
NE_{21}	1.00	1	17.28	97	1.00	1	4.75	47
NE_{22}	1.00	1	1.00	1	1.51	6	4.75	47
$COMB_{00,10,20}$	1.13	2	4.05	59	1.07	4	4.75	47
$COMB_{01,21,21}$	1.00	2	24.50	105	1.07	4	4.75	47
$COMB_{02,12,22}$	1.00	2	4.05	59	1.98	8	4.75	47

case, the channel following the Voice channel generates the largest buffer size. For the COMB*ab* case, the channel shows similar size to the NE*ab* case following Voice channel.

CHAPTER 7. RESULTS ANALYSIS

The simulation results of the RSM-TP show that the designed protocol can be implemented with a reasonable amount of overhead in the network. In this chapter, the simulation results are analyzed considering network bit error rate, and the related future research is suggested.

Network Overhead

The overhead of the designed synchronization protocol is the increased packet size and the additional SYNC packets. For the DT-TPDU, 48 more bits are used in the variable header part. The SYNC packet has 208 bits of header and data, and the ACK packet has 104 bits. Assume only Idata channel generates acknowledgement for every data packet, and no error is occurred. Let

- i: Data type (0: Voice, 1: Video, 2: Sdata, and 3: Idata).
- D_i: Average packet size of channel i (bits).
- R_i: Average packet rate of channel i (packets/s).
- T_{sync}: Average synchronization interval (secs).

The total traffic rate in the network is

$$G = D_0R_0 + D_1R_1 + D_2R_2 + (D_3 + 104)R_3 \tag{7.1}$$

Table 7.1: Mean End-to-end Delay Increased by Synchronization Protocol

Synchronization Type	Voice (%)	Video (%)	Sdata (%)
$EQUAL$	0.00	92.59	11.71
NE_{00}	27.32	0.00	0.00
NE_{01}	0.00	729.63	0.00
NE_{02}	0.00	0.00	140.54
NE_{10}	19.07	0.00	0.00
NE_{11}	0.00	303.70	0.00
NE_{12}	0.00	0.00	52.25
NE_{20}	37.63	0.00	0.00
NE_{21}	0.00	500.00	0.00
NE_{22}	0.00	0.00	59.46
$COMB_{x0}$	27.32	92.59	11.71
$COMB_{x1}$	0.00	718.52	11.71
$COMB_{x2}$	0.00	92.59	109.91

The network overhead (OH) increased by each synchronization function is

$$OH_{EQUAL} = \frac{48(R_0 + R_1 + R_2)}{G} \tag{7.2}$$

$$OH_{Non-EQUAL} = \frac{208/T_{sync}}{G} \tag{7.3}$$

$$OH_{COMB} = \frac{48(R_0 + R_1 + R_2) + 208/T_{sync}}{G} \tag{7.4}$$

In the designed protocol, $G = 113.22Mbps$, $OH_{EQUAL} = 0.58\%$, $OH_{Non-EQUAL} = 0.012\%$, $OH_{COMB} = 0.592\%$.

The end-to-end delay increased by each synchronization function is shown in Table 7.1. The video packets generate the most significant increase of delay for synchronization. For the worst case, the end-to-end delay is above 9 ms. This is still less than the maximum allowable jitter for Video channel in [19]. Also for the same data type, the packet following the fast arriving Video packet shows the least increase of end-to-end delay. For COMB$_{xa}$ case, the channels except a give the same

Table 7.2: Packet Errors of Sdata Channel

Bit Error Rate	1×10^{-6}	1×10^{-5}	1×10^{-4}
Packet Error Rate	9.25×10^{-4}	1.288×10^{-2}	1.1×10^{-1}
Number of Errors	1	16	134

end-to-end delay as in EQUAL case, and channel *a* gives similar end-to-end delay as in Non-EQUAL case whose first channel is Voice.

Network Errors

The network bit error rate is assumed to be 1×10^{-6}, in which case the packet errors of the Voice and Video channels are zero during the 10 second simulation time. The end-to-end delays and the receive contiguous list sizes are measured with higher bit error rates: 1×10^{-5} and 1×10^{-4}. With those error rates the SDATA channel shows increased packet error rate. All the Voice packets and the Video packets are still accepted. The measured packet error rate and the number of errors of Sdata channel are shown in Table 7.2.

The EQUAL and COMB synchronization are simulated for increased network bit error rate. Table 7.3 and Table 7.4 show the end-to-end delay and the size of RCL for each network error rate. The average end-to-end delays slightly increase as the network error rate increases. The lost packets result in retransmission of Sdata and Idata, and generate synchronization errors. The high-rate synchronization errors also generate increased maximum end-to-end delays up to near 20 msecs for 10^{-5} and 10^{-4} bit error rates. Current high-speed network has usually very low bit error rate down to 10^{-9} errors per bit or less which keeps the lost packets as minimum for acceptable maximum delay. The buffer sizes are not much affected by the

Table 7.3: End-to-End Delay with Higher Network Error Rate (10^{-3} secs)

Sync. Type	Data Type	$ber = 10^{-6}$		$ber = 10^{-5}$		$ber = 10^{-4}$	
		Mean	Max	Mean	Max	Mean	Max
EQUAL	Voice	1.94	4.89	2.24	17.25	2.25	16.58
	Video	0.52	5.75	0.84	17.03	0.83	16.81
	Sdata	1.24	4.13	1.48	17.07	1.44	15.61
COMBx0	Voice	2.47	5.64	2.52	10.01	2.61	14.73
	Video	0.52	5.27	0.58	7.13	0.74	15.55
	Sdata	1.24	4.02	1.28	6.59	1.40	13.18
COMBx1	Voice	1.94	4.17	2.00	7.01	2.14	14.06
	Video	2.21	8.98	2.22	12.05	2.23	18.42
	Sdata	1.24	4.02	1.28	6.59	1.40	13.17
COMBx2	Voice	1.94	4.19	2.00	7.01	2.14	14.06
	Video	0.52	5.27	0.58	7.13	0.74	15.55
	Sdata	2.33	8.50	2.34	10.99	2.41	17.24

Table 7.4: Receive Contiguous List Size with Higher Network Error Rate

Sync. Type	Data Type	$ber = 10^{-6}$		$ber = 10^{-5}$		$ber = 10^{-4}$	
		Mean	Max	Mean	Max	Mean	Max
EQUAL	Voice	1.00	2	1.00	2	1.01	2
	Video	4.05	62	4.07	65	4.68	67
	Sdata	1.07	4	1.08	6	1.07	6
COMBx0	Voice	1.13	2	1.13	2	1.12	2
	Video	4.05	59	4.04	62	4.71	68
	Sdata	1.07	4	1.06	3	1.06	5
COMBx1	Voice	1.00	2	1.00	2	1.01	2
	Video	24.50	105	24.00	107	22.89	109
	Sdata	1.07	4	1.06	3	1.06	5
COMBx2	Voice	1.00	2	1.00	2	1.01	2
	Video	4.05	59	4.04	62	4.71	68
	Sdata	1.98	8	1.93	8	1.81	8

network error rate. For Voice, Video packets, the increased error rate is not sufficient to generate the enough packet errors that affect the buffer size. When a packet error occurs, the following packets are inserted into the receive buffer (RL) and are waiting for sequencing function by the transport protocol, which results in slight decrease of the buffer size (RCL) that is used for synchronization.

Design Consideration

The designed protocol performs synchronization for three real-time data channels. This design uses simplified timing consideration for the protocol function. The processing time inside the protocol function has to be made minimum to implement high-speed protocol with synchronization by using global shared memory and multiprocessor implementation. Using the emerging high-speed implementation of the current transport protocol, sufficient processing speed can be expected. The transport protocol uses a timer for each synchronization process. The efficient implementation of the timer is one of the important issues for the synchronization protocol design.

Future Work

The RSM-TP protocol in this research is intended for multimedia packet synchronization between several independent channels in the communication network. It shows a possible extension of the current OSI-TP with reasonable overhead. From this design, there are several future research areas that can make the design more flexible and reliable.

- Multiple packet synchronization with any synchronization combination.

- Recursive packet synchronization in which the SYNC packets are also pointed by higher level synchronization scheme.
- High level stream or frame synchronization.
- Multiple node synchronization that can handle more than two communication nodes.
- Operating system implementation.
- Hardware implementation.
- Analytical modeling for synchronized channels.
- Network synchronization monitor design.

CHAPTER 8. CONCLUSIONS

High speed communication networks make it possible to communicate with real-time data which typically contain voice and video data. Such real-time data components usually have temporal relations with one another. The most frequent timing relation resides in the video data that carry their corresponding voice data. To transfer meaningful data to the destination, the timing relation must be preserved when the data are received.

In this research, a communication protocol with a packet synchronization function has been designed. Because the synchronization protocol is performed between the two end users, it is designed on the transport layer and uses the OSI-TP as its base protocol for possible extension. Also, the synchronization protocol is designed to introduce minimum overhead onto the current network traffic. To minimize the overhead and complexity, the timing relationships between packets are categorized into three types: 1) EQUAL, 2) Non-EQUAL, and 3) COMBINED. The EQUAL synchronization has the simplest timing relation that must be maintained for the destination users, so that the packets of several data channels are delivered to their transport service users within a specified time. It is implemented by modifying current DT-TPDU of OSI-TP without using any other packets or additional channels. The Non-EQUAL synchronization is used to specify the relative delivery time between packets on the

same or different channels. A synchronization channel is set up to carry the SYNC packet (SYNC-TPDU) which contains the timing information for the specific packets to be synchronized. The SYNC-TPDU has a header part and a data part like the other TPDU. The header of SYNC-TPDU is newly defined in the protocol, the data part carries the necessary synchronization information. The SYNC-TPDU can contain as much information as necessary for possible future extension. It has versatile synchronization functions, but introduces overhead to set up an additional channel. The COMB synchronization has both EQUAL and Non-EQUAL functions, and it performs the two types of synchronization at the same time. With proper use of the COMB synchronization, several types of synchronization of any channel combination can be achieved with reasonable network and protocol overhead. The pure network overhead introduced by the complex COMB synchronization is 0.592% of the total network traffic. The simulation results show some variation for the end-to-end delay, especially for the Video data which has the fastest data rate. The buffer for the synchronization function is needed for the Video channel. The buffer becomes empty as soon as either normal synchronization or error occurs. Therefore, the buffer size is mainly affected by the packet errors of synchronized channels and the synchronization time-out value.

The fast deceasing network error rate makes it possible to design more efficient synchronization protocol without considering lost packets of the real-time Voice, Video, and with minimum concern of the real-time data channels. With 10^{-4} bit error rate, all the video and voice packets are acceptable with given specification. But the maximum end-to-end delay of each data channel gives untolerable value with synchronization function. Since the emerging high-speed protocols are expected to

have much lower network error rate, there is little packet loss that makes the synchronization process difficult. This protocol is designed based on the current OSI Transport Protocol, which makes it possible to implement this design on the emerging high-speed transport protocol that is based on the implementation of the OSI protocol.

BIBLIOGRAPHY

[1] Dietmar B. Hehmann, Michael G. Salmony, and Heinrich J. Stüttgen. "Transport Services for Multimedia Applications on Broadband Networks." *Computer Communications,* Vol. 13, No. 4, (May 1990), 197-203.

[2] Luis Oroco Barbosa, and Nicolas D. Georganas. "Multimedia Services and Applications." *European Transactions on Telecommunications and Related Technologies,* Vol 2, No. 1, (January - Febuary 1991), 5-19.

[3] Cosmos Nicolau. "An Architecture for Real-Time Multimedia Communication Systems." *IEEE Journal on Selected Areas in Communications,* Vol 8, No. 3, (April 1990), 391-400.

[4] Doug Shepherd and Michael Salmony. "Extending OSI to Support Synchronization Required by Multimedia Applications." *Computer Communications,* Vol. 13, No. 7, (September 1990), 399-406.

[5] Gerd Schümann and Uwe Holzmann-Kaiser. "Distributed Multimedia Information Handling and Processing." *IEEE Network Magazine,* (November 1990), 23-31.

[6] Andrew S. Tanenbaum. *Computer Networks.* 2nd Edition. Prentice Hall, Englewood Cliffs, NJ, 1988.

[7] Thomas F. La Porta and Mischa Schwartz. "Architectures, Features, and Implementation of High-Speed Transport Protocols." *IEEE Network Magazine,* (May 1991) 14-22.

[8] Mirjana Zafirovic-Vukotic and Ignas G. Niemegeers. "Multimedia Communication Systems : Upper Layers in the OSI Reference Model." *IEEE Journal on Selected Areas in Communications,* Vol. 10, No. 9, (December 1992), 1397-1402.

[9] William Stallings *Handbook of Computer-Communications Standard Volume 1 : The Open Systems Interconnection (OSI) Model and OSI-Related Standards,* Macmillan, New York, 1987.

[10] Martina Zitterbart. "High Speed Transport Components." *IEEE Network Magazine,* (January 1991), 54-63.

[11] David R. Chreiton, and Carey L. Williamson. "VMTP as the Transport Layer for High-Preformance Distributed Systems." *IEEE Communications Magazine,* June 1989, 37-44.

[12] Greg Chesson. "XTP/PE Design Considerations." pp. 27-33 in *IFIP WG6.1/WG6.4 International Workshop on Protocols for High-speed Networks,* Zurich, Switzerland, May 9-11, 1989.

[13] H. Abu-Amara, T. Balraj, T. Barzilai, and Y.Yemini. "PSi : A Silicon Compiler for Very Fast Protocol Processing." pp. 181-195 in *IFIP WG6.1/WG6.4 International Workshop on Protocols for High-speed Networks,* Zurich, Switzerland, May 9-11, 1989.

[14] Zygmunt Haas. "A Communication Architecture for High-Speed Networking." pp. 433-441 in *Proceedings in IEEE INFOCOM '90,* San Francisco, USA, June 3-7, 1990.

[15] Sylvie Dupuy, Wassim Tawbi, and Eric Horlait. "Protocols for High-Speed Multimedia Communications Networks." *Computer Communications,* Vol. 15, No. 6, (July/August 1992), 349-358.

[16] Dario Giarrizzo, Matthias Kaiserswerth, and Thomas Wicki, and Robin C. Williamson. "High-Speed Parallel Protocol Implementation." pp. 165-180 in *IFIP WG6.1/WG6.4 International Workshop on Protocols for High-speed Networks,* Zurich, Switzerland, May 9-11, 1989.

[17] Niraj Jain, Mischa Schwartz, and Theodore R. Bashkow. "Transport Protocol Processing at GBPS Rates," pp. 188-199 in SIGCOMM 1990 *Communications Architecture & Protocols,* Philadelphia, PA, September 24-27, 1990.

[18] Nigel A. Davies and John R. Nicol. "Technological Perspectives on Multimedia Computing." *Computer Communications,* Vol. 14, No. 5, (June 1991), 260-272.

[19] Dietmar B. Hehmann, Michael G. Salmony, and Heinrich J. Stüttgen. "High-speed Transport Systems for Multimedia Applications." pp. 303-321 in *IFIP WG6.1/WG6.4 International Workshop on Protocols for High-speed Networks,* Zurich, Switzerland, May 9-11, 1989.

[20] Ralf Steinmetz. "Synchronization Properties in Multimedia Systems." *IEEE Journal on Selected Areas in Communications,* Vol. 8, No. 3, (April 1990), 401-412.

[21] P. Venkat Rangan, Srinivas Ramanathan, Harrick M. Vin, and Thomas Kaeppner. "Techniques for Multimedia Synchronization in Network File Systems." *Computer Communications,* Vol. 16, No. 3, (March 1993), 168-176.

[22] Imrich Chlamtac, and Aura Ganz. "Design and Analysis of Very High-Speed Network." *IEEE Journal on Selected Areas in Communications,* Vol 36, No. 3, (March 1988), 252-262.

[23] Thomas D. C. Little and Arif Chafor. "Synchronization and Storage Models for Multimedia Objects." *IEEE Journal on Selected Areas in Communications,* Vol. 8, No. 3, (April 1990), 413-427.

[24] Wu-Hon F. Leung, Thomas J. Baumgartner, Yeou H. Hwang, Mike J. Morgan, and Shi-Chuan Tu. "A Software Architecture for Workstations Supporting Multimedia Conferencing in Packet Switching." *IEEE Journal on Selected Areas in Communications,* Vol 8, No. 3, (April 1990), 380-390.

[25] Marshall T. Ross. "Transition and Coexistence Strategies for TCP/IP to OSI." *IEEE Journal on Selected Areas in Communications,* Vol 8, No. 1, (January 1990), 57-66.

[26] Martina Zitterbart. "High-Speed Protocol Implementations Based on a Multiprocessor Architecture." pp. 151-164 in *IFIP WG6.1/WG6.4 International Workshop on Protocols for High-speed Networks,* Zurich, Switzerland, May 9-11, 1989.

[27] David D. Clark, Van Jacobson, John Romkey, and Howard Salwen. "An Analysis of TCP Processing Overhead." *IEEE Communications Magazine,* (June 1989), 23-29.

[28] Alain J. Cohen. *OPNET (Optimized Network Engineering Tools), Modeling Manual, Simulation Kernal Manual, Tool Operating Manual, External Interface Manual,*. 1991, Mil3 Inc., Washington D.C., 1991

[29] Alain J. Cohen. *OPNET (Optimized Network Engineering Tools), Base Models Manual,* 1991, Mil3 Inc., Washington D.C., 1991

[30] Alain J. Cohen. *OPNET (Optimized Network Engineering Tools), Example Designs,* 1991, Mil3 Inc., Washington D.C., 1991

[31] International Standard Organization "ISO Transport Protocol Specification ISO DP 8073." *Request for Comments 9505,* April, 1984.

[32] Leonard Kleinrock. *Queueing Systems, volumn 1, Theory,* John Wiley & Sons, New York, NY, 1975.

[33] Jeremiah F. Hayes. *Modeling and Analysis of Computer Communications Network,* Plenum Press, New York, NY 1984.

APPENDIX RSM-TP FLOWCHARTS

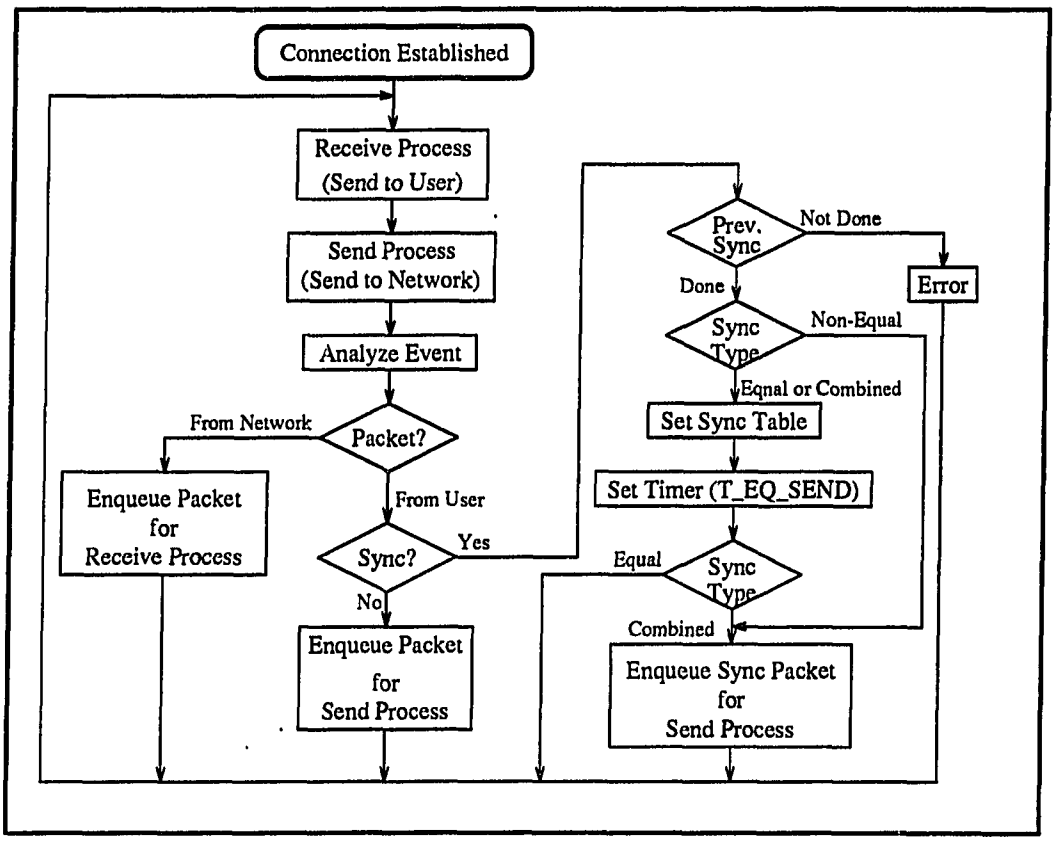

Figure A.1: RSM-TP Main Flowchart

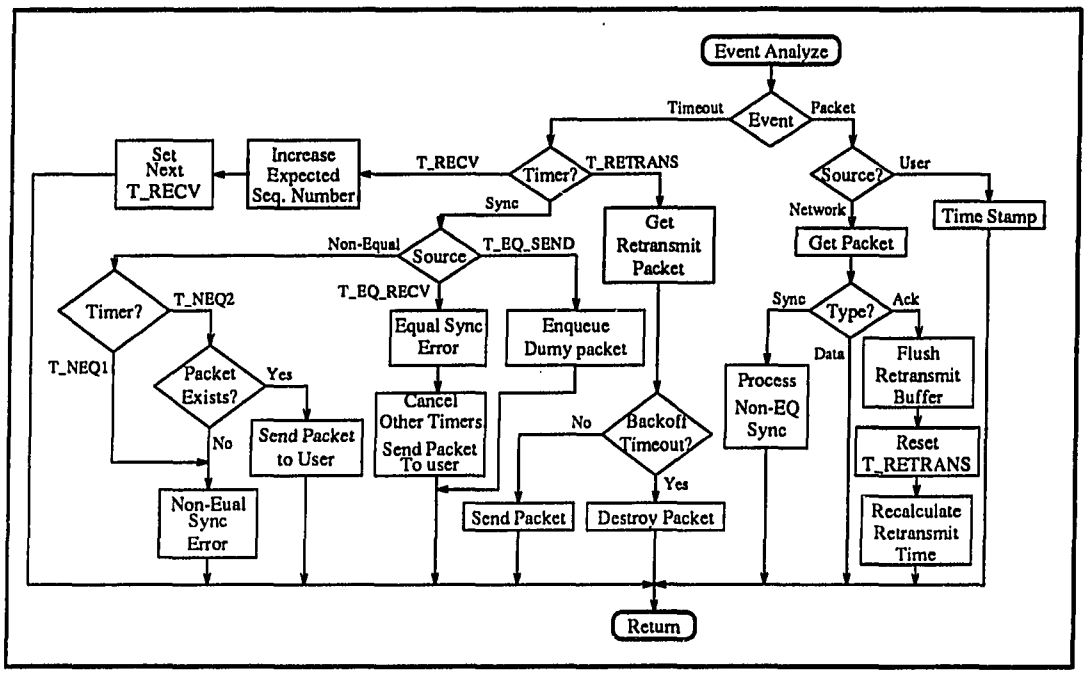

Figure A.2: RSM-TP Event Analysis

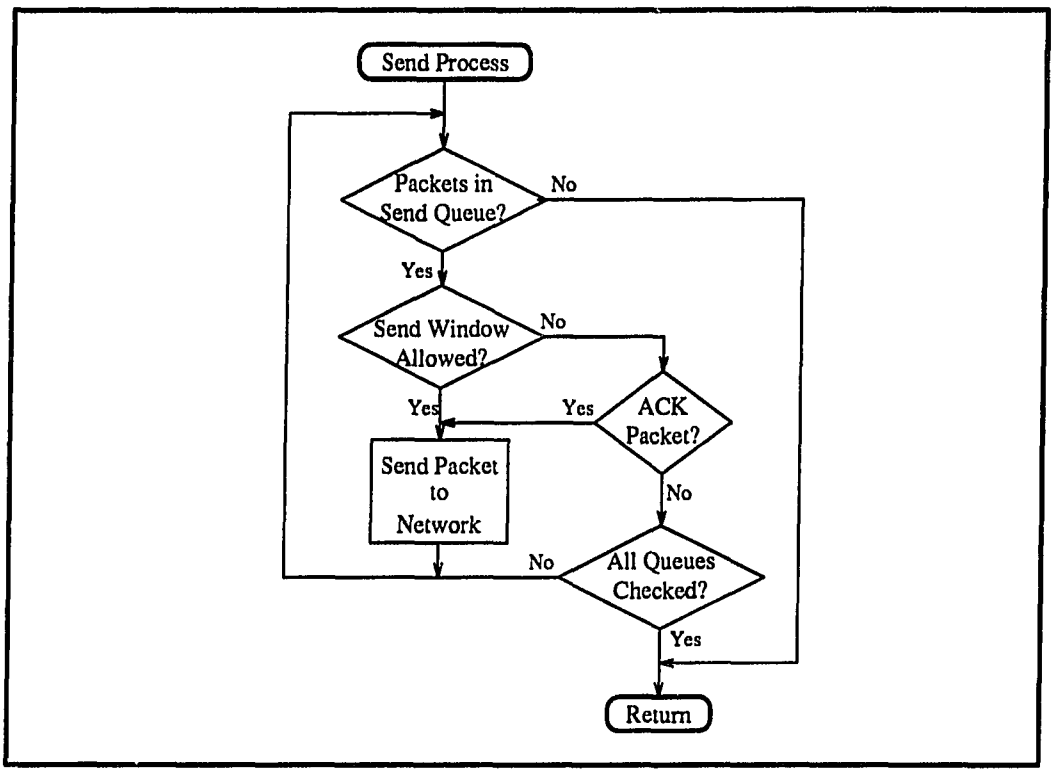

Figure A.3: Send Request Process

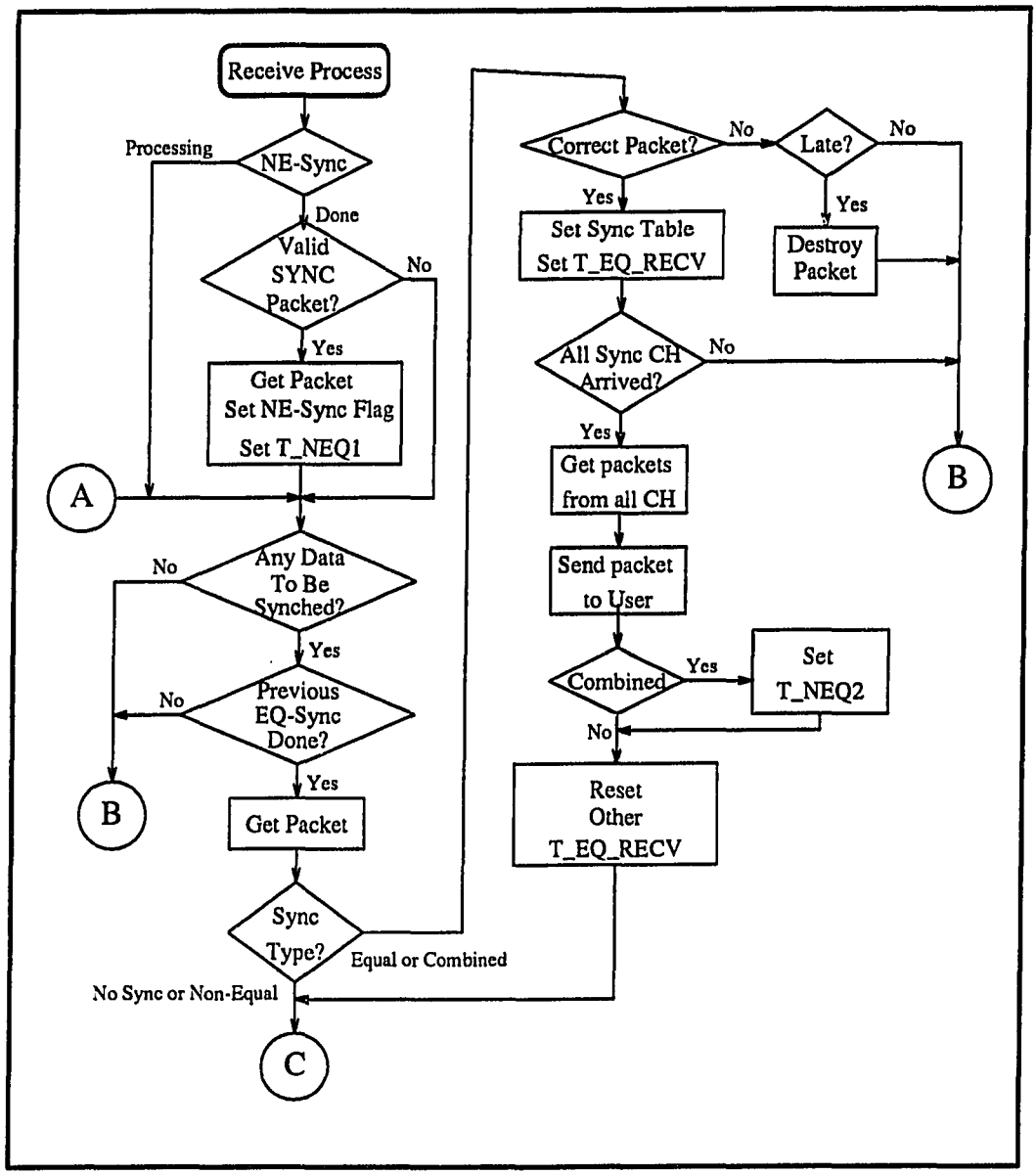

Figure A.4: Receive Request Process I

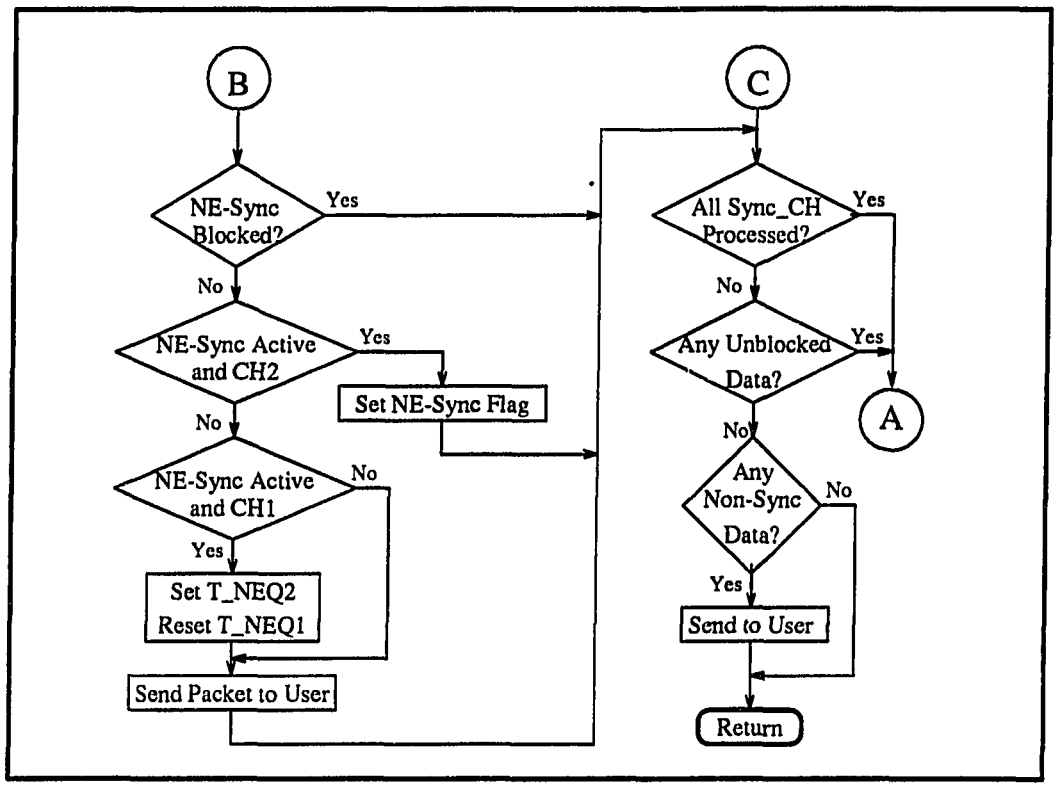

Figure A.5: Receive Request Process II

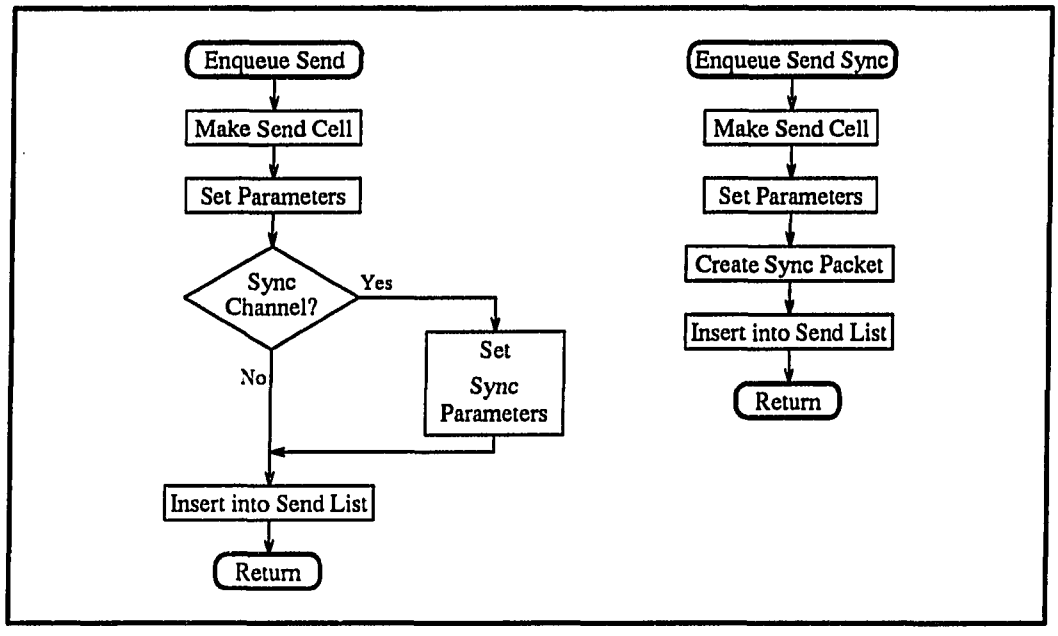

Figure A.6: Enqueue Send

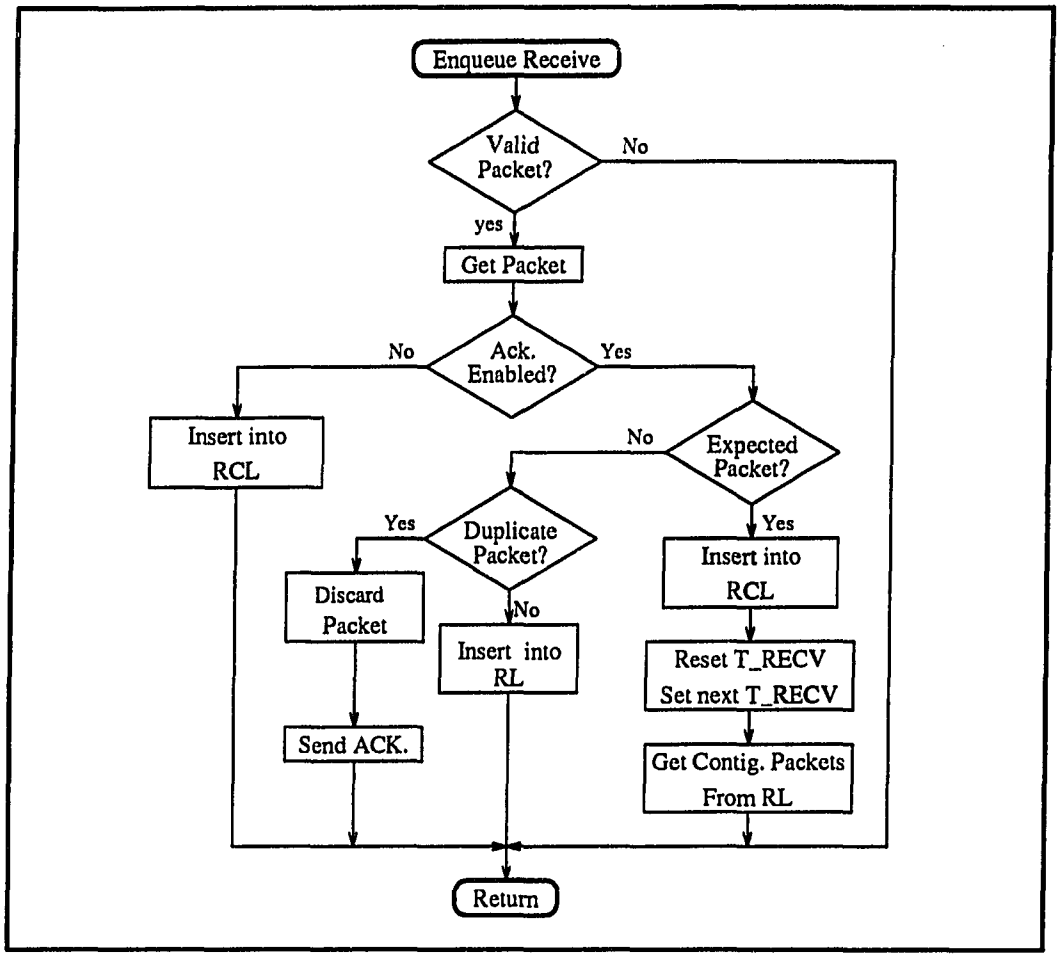

Figure A.7: Enqueue Receive

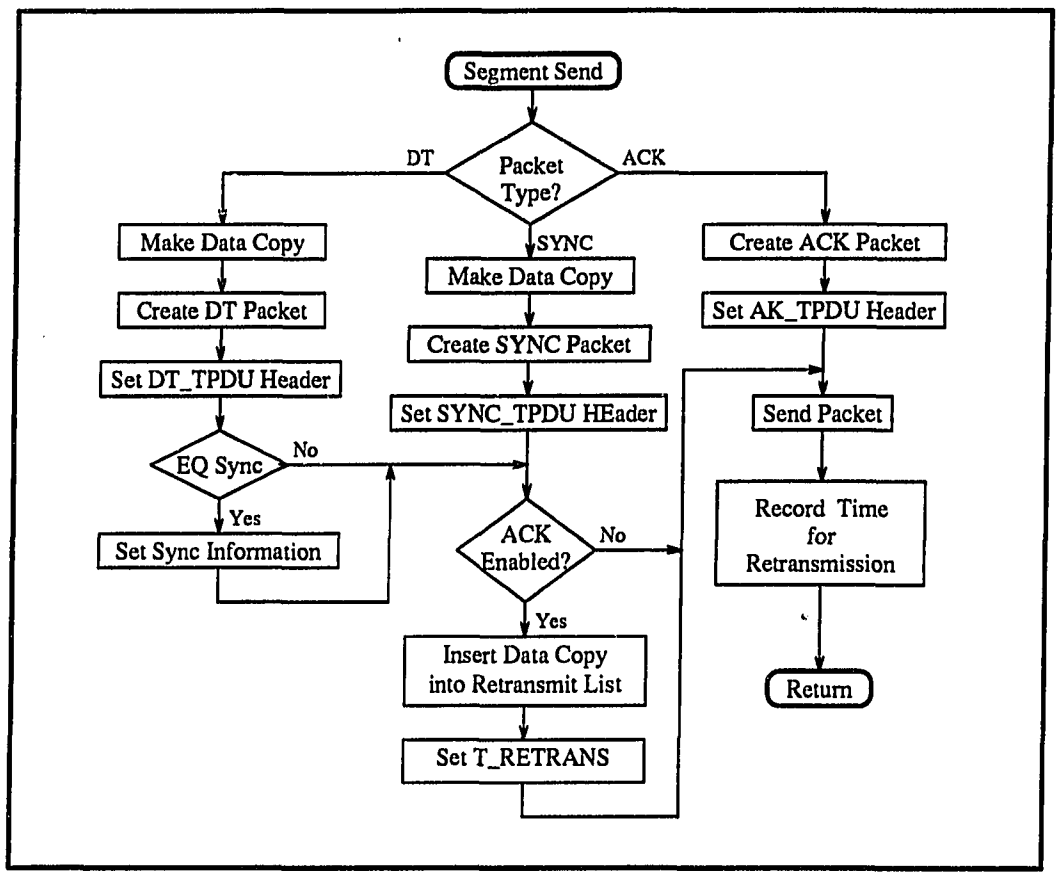

Figure A.8: Segment Send

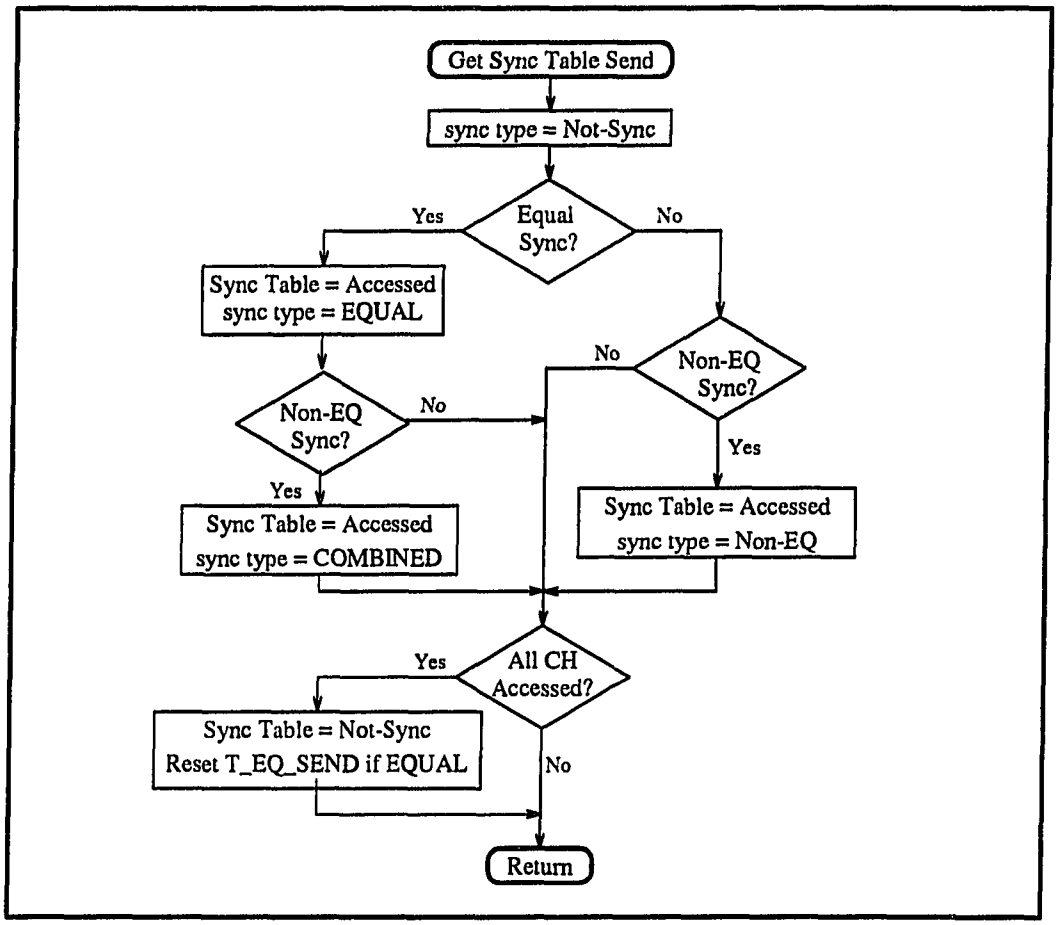

Figure A.9: Get Sync. Table Send

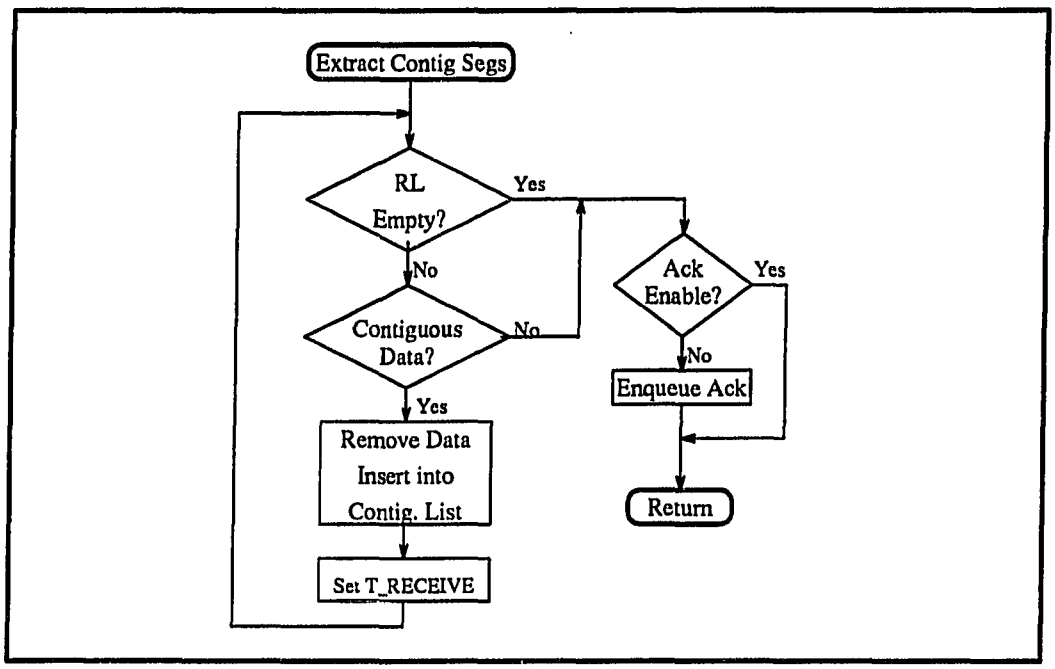

Figure A.10: Extract Contig. Segment